

Homomorphic Functional Encryption: Trustless Key Derivation for Functional Encryption*

Camille Nuoskala¹, Hossein Abdinasibfar^{1,†}, Mélina Hadjeres¹ and
Antonios Michalas^{1,†}

Tampere University, Tampere, Finland,
`{firstname.lastname}@tuni.fi`

Abstract. In this paper, we study the trust assumptions underlying key generation in functional encryption (FE) schemes, as well as the information leakage that can arise from the use of functional decryption keys. FE schemes typically rely on a trusted key curator who holds the master secret key and derives functional decryption keys. This requirement makes the curator a central point of trust and a particularly sensitive target for compromise. We show how homomorphic encryption (HE) can be used to mitigate this problem by allowing FE key-derivation algorithms to be evaluated homomorphically. Building on prior work on function-blind key derivation [CHL20], we extend this approach to a three-party deployment in which the key curator does not hold the FE master secret key in the clear. The resulting construction preserves function blindness while protecting the master secret key from the curator, thereby reducing the trust assumptions associated with conventional FE key management. Additionally, to test the applicability of our approach, we made a proof-of-concept implementation and ran a series of benchmarks. Finally, as a way to support open science and reproducible research, we make our code publicly available.

Keywords: Function Blindness, Functional Encryption, Homomorphic Encryption, Homomorphic Evaluation

1 Introduction

Since the multiple breakthroughs in Public-Key Encryption (PKE) [DH76, RSA78, McE78], a new need has unfolded in cryptography, namely the need to continue using encrypted data. Though Rivest *et al.* originally theorized the field in 1978 [RD78], it was the work of Gentry in 2009 [Gen09] that signaled its maturity and helped us realize that Homomorphic Encryption (HE) could indeed offer the best of both worlds and guarantee privacy, all while allowing the use and analysis of gathered data. Though this first milestone took more than 30 years, the following 15 years gave birth to a plethora of schemes supporting a range of data proceeding from *integers* [Bra12, FV12, BGV12] to *complex numbers* [CKKS17] and handling basic functions like addition or multiplication to the most complex arithmetic circuits [CGGI20].

All-or-Nothing issue in HE. Similar to other PKE schemes, HE schemes encrypt all data under the same public key pk ; consequently, the owner of the corresponding secret key sk can decrypt and get access to all of the data and/or computations' results. This presents an important challenge in access control, where the data owner wishes to

*This work was funded by the PiQASO EU research project (No. 101190366).

[†]Corresponding author

share only the results of the computation with other parties. The recent development of Functional Encryption (FE) [SW05, BSW11] provides a solution to this issue.

The fundamental difference with HE is that FE does not rely on a single secret key but uses a key generation algorithm that outputs multiple decryption keys with remarkable capabilities. In practice, each functional decryption key dk_f is associated with a function f . In contrast to traditional cryptosystems, using dk_f on a ciphertext $\text{Enc}(x)$ does *not* recover x but a function $f(x)$, hence keeping the value x private. This structure opens a wide range of possibilities, particularly in cloud-based applications, where multiple users store large volumes of data in remote, possibly corrupted entities.

Privacy & Trust in Functional Encryption. As we have seen in many different contexts, privacy is a large and very nebulous target that is hard to approach directly. This challenge extends to FE, where, despite the remarkable capabilities of the schemes, there is a fundamental issue hiding beneath its surface. More precisely, as in many cryptographic schemes there is the need for a trusted entity – one of the big hurdles that cryptographers have wrestled with for decades. Many FE schemes rely on the existence of a trusted entity, often called a Key Generator (KG) or Key Curator (KC). This entity is responsible for generating a master public and secret key pair (msk, mpk) during the setup phase. Additionally, it has the power to generate any functional decryption key dk_f for a function f , upon request.

This assumption poses two main privacy issues: (1) The level of trust placed in the curator may be unrealistic. This is because the curator can generate a decryption key for any function and any set of ciphertexts, hence making it a master entity that can eventually decrypt all data and breach user privacy at will. (2) We identified that FE schemes naturally suffer from the important, till now overlooked, problem of *function blindness*. That is, ensuring that a curator who is performing a functional decryption generation will output the correct result *without* learning anything about the computed function. In other words, when the curator receives a request from an analyst asking to get a functional decryption for a function f , the curator should process this in a blind way. This means that:

1. The curator must be able to generate a correct dk_f in a blind way (i.e. without knowing that this key corresponds to function f).
2. If, at a later point, the curator receives another request asking to generate again a functional decryption key for the same function f , they should not be able to determine that these two requests correspond to the same function.

Current Solutions. The need for a trusted entity who will be managing keys is a well-known issue in the FE literature. Furthermore, while a trusted entity is a fundamental assumption for a slew of applications and cryptographic constructions, in many cases undermines the overall security and privacy of the underlying schemes, thus pushing us towards building our mathematical edifices on falsehoods. Researchers have made significant efforts to address this limitation by finding new ways to look at this. Existing solutions include Decentralized [CDSG⁺18a, ABKW19] and Dynamic Decentralized [CDSG⁺20, NPS25] FE, and Registration-Based FE [GHMR18, FFM⁺23, DPY23], which are considered Multi-Party FE (MPFE) [AGT21]. These approaches primarily shift the generation process of master keys from KC to the data owners or, in some cases, to distributed (yet trusted) authorities. This change means that the KC will no longer hold the master secret keys, but only some public information. While these solutions offer important advantages, they also come with certain limitations and are based on very specific system models. In contrast, our solution, does not have the need for that as it is agnostic to the underlying FE construction. In other words, we show how an FE scheme that is using a trusted KC

can remove the levels of trust required by that entity¹.

1.1 Contributions

The primary contribution of this work is the design of Homomorphic Functional Encryption (HFE) – a generic construction that removes the need for a trusted entity while inherently supporting the function blindness feature within traditional FE. To achieve this, our construction utilizes HE to evaluate the decryption key derivation circuit for FE schemes, thereby transforming traditional FE schemes into more secure schemes that can outsource their computation to *any* service provider. In contrast to traditional FE schemes, HFE separately encrypts the master secret key and requested function, allowing the curator to evaluate the FE key-derivation circuit without holding either value in the clear.

High-Level Overview. To initiate our construction, the parties (i.e., user, curator, analyst) agree on an HE and an FE scheme. The user generates locally the master keys msk, mpk while the analyst has his own unique keys sk and pk . The encryption of the data is done as in a classic FE setup: The user encrypts his data using the master public key mpk . The main difference lies in the way functional decryption keys are generated. Instead of running the key derivation algorithm using msk and a function f , the curator only receives their encryption under the HE scheme, that is c_{msk} and c_f . More precisely, the user who owns msk , encrypts it homomorphically with pk and sends the result c_{msk} to the curator. At a later point, when an analyst wishes to calculate a function f on a set of data (ciphertexts), she encrypts the function f homomorphically with pk and sends the result c_f to the curator. Following, the curator runs the homomorphic *evaluation* algorithm on the key derivation circuit. The output of this computation is a homomorphic encryption of the decryption key dk_f , which can only be decrypted by the party holding the corresponding secret key sk , for example, a data analyst. The final result is obtained by running the classic FE decryption algorithm using the decryption key dk_f on encrypted data given by the user.

Contributions. The core contributions of this paper can be summarized as follows:

- C1.** We define HFE as a generic extension of function-blind FE key derivation, with separate encryption of the master secret key and requested function in a three-party model. Our construction improves trust of FE schemes as the key curator need not hold the master secret key in the clear, thereby reducing the trust placed in that party.
- C2.** We show how HFE can be used to create applications where no trusted entity is required. Hence, our construction provides a realistic solution to the long-standing issue of trusted entities, which has been bubbling under many of the existing FE schemes.
- C3.** We show how our construction naturally supports function-blindness – an important feature in FE that reduces the overall leakage of the underlying scheme.
- C4.** We present a real-world application of HFE, showing its applicability in realistic scenarios. HFE relies on a model of trust that does *not* require a trusted authority, presenting a strong alternative to existing Multi-Party Computation (MPC) techniques. We hope that this work will kick-start a period of greater trust and sustained research in the field of FE.
- C5.** We validate the practicality of our approach through a proof-of-concept implementation and extensive benchmarking. To support open science and reproducible research, we make our code publicly available.

¹Our construction is a generic construction that theoretically supports any FE key derivation to be evaluated under HE as long as the message and cipher space for the underlying key derivation function is supported by the chosen HE scheme.

2 Related Work

This article combines the two notions of HE and FE. Although they share a common base for allowing computations on encrypted data, they present diametrical divergence in their application. In this section, we try to provide a comprehensive overview of the state-of-the-art in both fields to provide readers with a clear view of the possibilities, the limitations, and the challenges they pose.

Homomorphic Encryption (HE): HE is a primitive which has been the object of active research since its definition by Rivest *et al.* [RD78]. Initially seen as a solution to the incompatibility between user privacy and the need to delegate data, it has since become essential in a plethora of scenarios, such as e-voting [HS00] or more recently in privacy-preserving machine learning [SPTP⁺21, TZR⁺22, MZ17], to cite some of the most trendy applications.

HE concept is to perform computations on encrypted data to produce a ciphertext that can be decrypted into the same result as if these computations had been performed on plain data. This process is called *evaluation*. Depending on the complexity of the function, its evaluation requires different classes of HE schemes. One of the most efficient is *leveled* or *somewhat* homomorphic encryption (SHE). In this case, a function is evaluated up to a certain multiplicative depth chosen beforehand. The most emblematic SHE schemes are BGV [BGV12] and B/FV [Bra12, FV12], that allow very efficient encryption of integers with fast evaluation. However, in many *real-world* scenarios, supporting integers does not suffice. In 2017, Cheon *et al.* [CKKS17, KPP22] designed an ingenious method to support real numbers (more precisely, approximate arithmetic).

Though the latter three schemes offer efficient homomorphic evaluation, this is still restricted to functions with predetermined multiplicative depth, which breaks with the principle of HE to allow one to perform any kind of operations on the ciphertexts. This limitation can be overcome by a more powerful primitive, namely *fully* homomorphic encryption (FHE). Since its first realization by Gentry in 2009 [Gen09], FHE has been both very exciting due to its theoretical capabilities and frustrating due to its high computational requirements. A related direction is *hybrid homomorphic encryption* (HHE), which combines HE with other cryptographic schemes, such as symmetric encryption, to support hybrid encrypted computation [ANM26].

We believe that, at present, the most efficient, hence realistic FHE scheme is the Torus-based FHE (TFHE) scheme of Chillotti *et al.* [CGGI20], although there exist good alternatives relying on an extension of the schemes mentioned above, BGV / BFV [GV23] or CKKS [CHK⁺18]. The advantage of TFHE lies in its clever use of the mathematical structure of the *torus* to perform *bootstrapping* through the evaluation of look-up tables. The rest of this article presents limited background information on HE, besides its formal definition and properties, and recommends the survey by A. Acar *et al.* [AAUC18], which, despite being slightly outdated in terms of novel constructions, provides a comprehensive analysis of the HE schemes we rely on.

Despite its apparent malleability and continuous improvements, HE appears to have semantic issues that extend beyond technical limitations. In particular, it does not allow access control over the computations since it evaluates the ciphertext using a common evaluation key and decrypts it with a single secret key. This problem is often called the *all-or-nothing* paradigm.

Functional Encryption (FE): In contrast to HE, FE allows fine-grained control and selective computations over encrypted data. This technique has emerged as a generalization of PKE with the work of Boneh *et al.* [BSW11] in 2011. While the first constructions of FE allowed the computation of a function on a *single* ciphertext, recent works [GGG⁺14, BM20] introduced a general notion of multi-input FE (MIFE) that supports functions operating on a polynomial number of inputs. One of the most popular classes of FE is *inner*

product functional encryption (IPFE) [CDSG⁺18b, NPP23, ABG19, DOT18, AGRW17, CLT18] named after the functionality it supports. Among these schemes, we focus on the more recent class of *lattice-based* IPFE [AFV11, ADCM12, ABDGP15, ACGU20, PD21, MKMS22a, CRS⁺23] that has the advantage of being post-quantum secure and, in our case, in order to align with the aforementioned HE schemes relying on lattices and, more precisely, on the *learning with error* (LWE) hard problem. However, it is important to note that this may cause some limitations as far as the parameter choice and range of functions supported, according to the descriptions of E. Tairi and A. Ünal [TÜ24]. Finally, it is important to mention that, though most existing schemes revolved into standard functions like addition or inner product, recent research has extended the scope of the functionalities of FE, hence unleashing its full potential in a larger range of applications. An example is the conjoint work of Gay [Gay16] and Baltico *et al.* [BCFG17] on *quadratic* functional encryption (QFE), or the recent article of Nuoskala *et al.* addressing the problem of *partial decryption* [NAM24].

While the schemes offer a secure yet adaptive way to guarantee privacy while providing access to targeted computation, they have left a significant leakage on the functional query's side. Indeed, a party requesting the computation of a certain function has to send it publicly. This request not only reveals the intentions of the party sending the request but can also hint at what the encrypted data might contain. Let us take the example of a user encrypting their medical records on one side and a medical lab sending a request as a function f that reveals markers for cancer. If the data are not entirely leaked, an eavesdropper can be suspicious that the user is affected by a certain pathology, which is unacceptable if we intend to guarantee privacy to both parties. An answer to this problem is given by *function blindness*.

Function Blindness: The study of FE was first focused toward *message* privacy, followed by *function* privacy with the work of Shen *et al.* [SSW09]. In the years, many so-called *function-hiding* have been designed [BM23, KLM⁺18, BJK15, Tom19, WWSG23, BS15, BRS13a, BRS13b, FT18], consisting in masking the requested function from a user. As this notion is not sufficient to mask the function to every entity, Canard *et al.* [CHL20] introduced *blind functional encryption*, which **only** hides the function from the entity responsible for key derivation. To the best of our knowledge, this is the only article addressing this problem. Canard *et al.* [CHL20] already give a two-party protocol in which HE hides the function from the authority while the authority evaluates the FE key-generation circuit. Their construction therefore establishes function-blind key derivation. Our focus is complementary: HFE formalizes a three-party model in which the user owns and encrypts the master secret key, the analyst supplies the HE key and requested function, and the key curator receives encrypted inputs rather than msk or f in the clear. Under the stated correctness and non-collusion assumptions, this protects msk from the curator in addition to hiding f . HFE should therefore be viewed as a generalization of the trust model and interface, not as the first function-blind FE construction.

3 Background

This section provides the necessary mathematical background and definitions for understanding our article.

Notation: For $n \in \mathbb{N}^*$, $[n]$ is the set $\{1, \dots, n\}$. Vectors are written in bold lowercase letters, and ones denotes $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$, the inner product of two vectors $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$. For a set $\mathcal{Y}$, one uses $x \leftarrow^* \mathcal{Y}$ if x is sampled at random from $\mathcal{Y}$. Generally, we write $x \xleftarrow{\chi} (\mathcal{Y})$ if x is sampled from $\mathcal{Y}$ following a distribution χ . The image, or range, of a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is denoted $\text{Im}(f) \subseteq \mathcal{Y}$. The probability of an event $A \in \Omega$ is denoted $\mathbb{P}(A)$, where $\mathbb{P} : \Omega \rightarrow [0, 1]$ is a probability measure on a set Ω .

usually omitted for brevity. A (probabilistic) polynomial time algorithm $\mathcal{A}$, or (P)PT, is a (randomized) algorithm for which there exists a polynomial $P(x)$ s.t. for all input x , the running time of $\mathcal{A}(x)$ is bounded by $|P(x)|$. A function $f : \mathbb{N} \mapsto \mathbb{R}$ is said negligible if $\forall \delta \in \mathbb{N}, \exists x_0 \in \mathbb{N}$ such that $\forall x \geq x_0, f(x) < x^{-\delta}$. One denotes $\text{negl}(\cdot)$ an arbitrary negligible function.

Throughout the article, we often refer to the following parties: user (**U**), key curator (**KC**), and data analyst (**DA**). To avoid any misunderstanding while referring to them in the text, we use the pronouns *he* for a user, *it* for the curator, and *she* for the analyst.

3.1 Encryption Schemes

Definition 1 (Public-Key Encryption (PKE)). A Public-Key Encryption scheme for message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ consists of three polynomial time algorithms KGen, Enc, Dec defined as follows:

- **KGen** (1^λ). The key generation algorithm is a probabilistic algorithm. On the input of a security parameter λ , it outputs a pair of keys (sk, pk) , respectively the secret and public keys;
 - **Enc** (pk, x). The encryption algorithm is a probabilistic algorithm. On the input of a public key pk and a message m in the message space $\mathcal{M}$, it outputs a ciphertext c in the ciphertext space $\mathcal{C}$. One also writes $c \leftarrow^s \text{Enc}_{\text{pk}}(x)$;
 - **Dec** (sk, c). The decryption algorithm is a deterministic algorithm. On the input of a secret key sk and a ciphertext c , it outputs a message x . One also writes $\text{Dec}_{\text{sk}}(c) \rightarrow x$.
- Correctness:** For all parameters $\lambda \in \mathbb{N}$, $(\text{sk}, \text{pk}) \leftarrow^s \text{KGen}(1^\lambda)$ and ciphertext $c \leftarrow^s \text{Enc}_{\text{pk}}(x)$, $\mathbb{P}[\text{Dec}_{\text{sk}}(c) \neq x] = \text{negl}(\lambda)$.

The security of a PKE scheme relies on the fact that the encryption algorithm **Enc** is probabilistic. For well-chosen parameters, it is unlikely for two ciphertexts $c \leftarrow^s \text{Enc}_{\text{pk}}(x)$ and $c' \leftarrow^s \text{Enc}_{\text{pk}}(x)$ of the same plaintext to be equal. However, these are both valid encryptions of x and hence can be correctly decrypted. We call the set of such ciphertexts the *image* of $\text{Enc}_{\text{pk}}(x)$ and denote it as $\text{Im}(\text{Enc}_{\text{pk}}(x))$. More formally,

$$\text{Im}(\text{Enc}_{\text{pk}}(x)) = \{c \in \mathcal{C} \mid \text{Dec}_{\text{sk}}(c) = x\}$$

(Approximate) Homomorphic Encryption: Following, we provide the definition of HE. Since our homomorphic solution is parametrized by an arbitrary HE scheme, we decide to use a global definition that includes both *approximate* and *exact* HE schemes. We discuss the difference between these two paradigms later on.

Definition 2 (Homomorphic Encryption). A Homomorphic Encryption scheme HE for message space $\mathcal{M}_{\text{HE}}$, ciphertext space $\mathcal{C}_{\text{HE}}$ and class of functions $\mathcal{F}_{\text{HE}} : \mathcal{M}_{\text{HE}}^n \rightarrow \mathcal{M}_{\text{HE}}$, $n \in \mathbb{N}^*$, consists of four polynomial-time algorithms HE.KGen, HE.Enc, HE.Eval, HE.Dec defined as follows:

- **HE.KGen** (1^λ). The key generation algorithm is a probabilistic algorithm. On the input of a security parameter λ , it outputs a tuple of keys $(\text{sk}, \text{pk}, \text{evk})$, respectively the secret, public and evaluation keys;
- **HE.Enc** (pk, x, Δ). The encryption algorithm is a probabilistic algorithm. On the input of a public key pk , a message x in the message space $\mathcal{M}_{\text{HE}}$ and a scaling factor $\Delta \in \mathbb{N}$, it outputs a ciphertext c in the ciphertext space $\mathcal{C}_{\text{HE}}$. One also writes $c \leftarrow^s \text{HE.Enc}_{\text{pk}}(x, \Delta)$;
- **HE.Eval** ($\text{evk}, f, (c_1, \dots, c_n)$). The evaluation algorithm is a deterministic algorithm that, on input the evaluation key evk , a function $f \in \mathcal{F}_{\text{HE}}$ and n ciphertexts $c_1, \dots, c_n$, outputs a unique ciphertext c . One can also write $\text{HE.Eval}_{\text{evk}}(f, (c_1, \dots, c_n)) \rightarrow c$;
- **HE.Dec** (sk, c, Δ). The decryption algorithm is a deterministic algorithm that, on input the secret key sk and a ciphertext c , outputs a message x . One also writes $\text{HE.Dec}_{\text{sk}}(c) \rightarrow x$.

Correctness: For all parameters $\lambda \in \mathbb{N}$, $n \in \mathbb{N}^*$, $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{HE.KGen}(1^\lambda)$, $f \in \mathcal{F}_{\text{HE}}$ and ciphertexts $c_i \leftarrow \text{HE.Enc}_{\text{pk}}(x_i, \Delta)$, $x_i \in \mathcal{M}_{\text{HE}}$, $i \in [n]$,

$$\mathbb{P} \left[\text{HE.Dec}_{\text{sk}}(\text{HE.Eval}_{\text{evk}}(f, (c_1, \dots, c_n)), \Delta) \neq f(x_1, \dots, x_n) + \delta \right] = \text{negl}(\lambda)$$

where δ is a small value (or noise) that depends on the scaling factor Δ . If HE is such that δ always vanishes, HE is said to be *exact*. Otherwise, it is called *approximate*.

Remark 1 (Evaluation Algorithm). The name *homomorphic* encryption comes from the notion of *group homomorphism*, widely used in mathematics. In an HE scheme, the encryption algorithm is indeed a group homomorphism between the group $\mathcal{M}_{\text{HE}}$ and the group $\mathcal{C}_{\text{HE}}$, with their respective operations. The particularity of a homomorphism is that it preserves these operations. More precisely, an operation in $\mathcal{C}_{\text{HE}}$ is associated to an operation in $\mathcal{M}_{\text{HE}}$. This is what the evaluation algorithm is: $\text{HE.Eval}_{\text{evk}}(f, \cdot)$ is the corresponding function operating on $\mathcal{C}_{\text{HE}}^n$ to a function $f(\cdot)$ that operates on $\mathcal{M}_{\text{HE}}^n$. Informally, one could write $\text{HE.Eval}_{\text{evk}}(f, (c_1, \dots, c_n)) = \text{HE.Enc}_{\text{pk}}(f(x_1, \dots, x_n))$. As the function $f : \mathcal{M}_{\text{HE}}^n \rightarrow \mathcal{M}_{\text{HE}}$ returns a single plaintext $x = f(x_1, \dots, x_n)$, the corresponding function returns a single ciphertext $c = \text{HE.Enc}_{\text{pk}}(f(x_1, \dots, x_n))$ that can be decrypted using the decryption algorithm $\text{HE.Dec}_{\text{sk}}(c)$ to retrieve $x = f(x_1, \dots, x_n)$.

Most commonly used HE schemes are exact, including BGV [BGV12], B/FV [Bra12, FV12] and TFHE [CGGI20] for example, and cover most application cases using integer data. However, to perform homomorphic operations on *real* data, or more accurately floating-point arithmetic, it is useful to consider approximate HE. The first and most popular example of such scheme is CKKS [CKKS17]. In theory, the only difference is the addition of a small noise in the final result. This noise is implicitly introduced in the encryption algorithm as a *scaling factor*, which determines the number of digits, or *most significant bits*, contained in the results.

(Noisy) Functional Encryption: Now that we have defined the two classes of HE schemes that are relevant in the scope of our article, we move on with FE, the second encryption technique on which our construction relies. Similarly, we provide a comprehensive definition that includes the novel notion of *noisy* functional encryption [ZASH24].

Definition 3 (Functional Encryption). A functional encryption scheme FE for message space $\mathcal{M}_{\text{FE}}$, ciphertext space $\mathcal{C}_{\text{FE}}$ and class of functions $\mathcal{F}_{\text{FE}}$, consists of four PT algorithms FE.Setup, FE.KDer, FE.Enc, FE.Dec defined as follows:

- FE.Setup(1^λ). The setup algorithm is a probabilistic algorithm. On the input of a security parameter λ it outputs a (master) secret key msk and a (master) public key mpk ;
- FE.Enc(mpk, x). The encryption algorithm is a probabilistic algorithm. On the input of public key mpk and a message x in the message space $\mathcal{M}_{\text{FE}}$, it outputs a ciphertext c in the ciphertext space $\mathcal{C}_{\text{FE}}$. One also writes $c \leftarrow \text{FE.Enc}_{\text{mpk}}(x)$;
- FE.KDer(msk, f, χ). The key derivation algorithm is a deterministic algorithm. On the input of the secret key msk , a function $f \in \mathcal{F}_{\text{FE}}$ and a probability distribution χ , it outputs a decryption key dk_f . One also writes $\text{FE.KDer}_{\text{msk}}(f, \chi) \rightarrow \text{dk}_f$;
- FE.Dec(dk_f, c). The decryption algorithm is a deterministic algorithm that, on the input of a decryption key dk_f and a ciphertext c , outputs a result y . One also writes $\text{FE.Dec}(\text{dk}_f, c) \rightarrow y$.

Correctness: For all parameters $\lambda \in \mathbb{N}$, $(\text{msk}, \text{mpk}) \leftarrow \text{FE.Setup}(1^\lambda)$, $f \in \mathcal{F}_{\text{FE}}$, $\text{dk}_f \leftarrow \text{FE.KDer}_{\text{msk}}(f, \chi)$ and $x \in \mathcal{M}_{\text{FE}}$,

$$\mathbb{P} [\text{FE.Dec}_{\text{dk}_f}(\text{FE.Enc}_{\text{mpk}}(x)) \neq f(x) + \delta] = \text{negl}(\lambda)$$

where $\text{Im}(f) \ni \delta \leftarrow \chi$ is a (small) noise sampled during the key derivation FE.KDer . If χ vanishes everywhere, FE is called *exact*. Otherwise, FE is said to be *noisy*, and we write nFE . In this latter case, the key derivation algorithm is probabilistic, and we write $\text{dk}_f \leftarrow^s \text{FE.KDer}_{\text{msk}}(f, \chi)$.

Remark 2 (Functional Decryption Algorithm). Unlike traditional encryption schemes, the decryption algorithm does not necessarily output an element of the plaintext space, but the result of a function that takes a value into an arbitrary space. This is one of the advantages of FE over HE, since the latter exclusively supports the function defined on the message space.

4 Homomorphic Evaluation of FE

This section describes how one can homomorphically evaluate the functional circuits to build a correct HFE scheme. In particular, it indicates the condition under which to design such a scheme.

4.1 Evaluation of FE Functions

This section studies the conditions for homomorphic evaluation of the FE key generation and decryption circuits. We consider an arbitrary FE scheme as defined in [Definition 3](#). We first describe how to homomorphically evaluate the functional key derivation algorithm with encrypted functionality and the master secret key. This theorem is fundamental to the establishment of a secure and function-blind HFE scheme in [Section 5](#).

Theorem 1 (Homomorphic Functional Key Derivation). *Let HE be an HE scheme and FE be an FE scheme. Let $\text{FE.KDer} : (\text{msk}, f) \mapsto \text{FE.KDer}(\text{msk}, f)$ be the 2-ary key derivation function. We make the following assumptions:*

- For any parameter $\lambda \in \mathbb{N}$ and pair of keys $(\text{msk}, \text{mpk}) \leftarrow^s \text{FE.Setup}(1^\lambda)$, msk lies in the message space $\mathcal{M}_{\text{HE}}$ of HE;
- FE.KDer is in the class of functions $\mathcal{F}_{\text{HE}}$;
- There exists a representation of f such that f is in the plaintext space $\mathcal{M}_{\text{HE}}$.

Under these assumptions, the key derivation algorithm can be run homomorphically, that is for all parameters $(\lambda_1, \lambda_2) \in \mathbb{N}^2$, $(\text{sk}, \text{pk}, \text{evk}) \leftarrow^s \text{HE.KGen}(1^{\lambda_1})$, $(\text{msk}, \text{mpk}) \leftarrow^s \text{FE.Setup}(1^{\lambda_2})$, $c_{\text{msk}} := \text{HE.Enc}_{\text{pk}}(\text{msk})$, and $f \in \mathcal{F}_{\text{FE}}$:

$$\text{HE.Eval}_{\text{evk}}(\text{FE.KDer}, c_{\text{msk}}, \text{HE.Enc}_{\text{pk}}(f)) \in \text{Im}(\text{HE.Enc}_{\text{pk}}(\text{dk}_f)) \quad (1)$$

with overwhelming probability in λ_1, λ_2 .

Proof. Let $(\lambda_1, \lambda_2) \in \mathbb{N}^2$ be arbitrary security parameters, for HE and FE respectively. Let $f \in \mathcal{F}_{\text{FE}}$ be an arbitrary function in the functionality space of FE. Run $(\text{sk}, \text{pk}, \text{evk}) \leftarrow^s \text{HE.KGen}(1^{\lambda_1})$ and $(\text{msk}, \text{mpk}) \leftarrow^s \text{FE.Setup}(1^{\lambda_2})$. For the sake of brevity and readability, we use the following notations: $c_{\text{msk}} := \text{HE.Enc}_{\text{pk}}(\text{msk})$, $c_f := \text{HE.Enc}_{\text{pk}}(f)$, and $g := \text{FE.KDer}$. The first member of [Equation \(1\)](#) becomes $\text{HE.Eval}_{\text{evk}}(g, c_{\text{msk}})$, where $g \in \mathcal{F}_{\text{HE}}$ by assumption. It follows:

$$\begin{aligned} \text{HE.Dec}_{\text{sk}}(\text{HE.Eval}_{\text{evk}}(g, c_{\text{msk}}, c_f)) = \\ g(\text{HE.Dec}_{\text{sk}}(c_{\text{msk}}), \text{HE.Dec}_{\text{sk}}(c_f)) \end{aligned}$$

(with overwhelming probability) because HE is correct. Hence,

$$\begin{aligned} \text{HE.Dec}_{\text{sk}}(\text{HE.Eval}_{\text{evk}}(g, c_{\text{msk}}, c_f)) \\ = \text{FE.KDer}(\text{msk}, f) = \text{dk}_f \end{aligned}$$

In other words, $\text{HE.Eval}_{\text{evk}}(g, c_{\text{msk}})$ is a valid encryption of dk_f , that is

$$\text{HE.Eval}_{\text{evk}}(g, c_{\text{msk}}, c_f) \in \text{Im}(\text{HE.Enc}_{\text{pk}}(\text{dk}_f))$$

with overwhelming probability in λ_1, λ_2 . $\square$

The main goal of this article is to address the privacy leakages that the generation of the decryption key causes. In [section 7](#), we utilize [Theorem 1](#) to build a protocol that both narrows the trust in the key curator and achieves function-blindness. However, the possibilities offered by HE are not limited to evaluating the key derivation algorithm, and our result can be extended to the decryption algorithm, as shown below.

Theorem 2 (Homomorphic Functional Decryption). *Let HE be an HE scheme and let $(\lambda_1, \lambda_2) \in \mathbb{N}^2$ be their respective security parameters. Let $\text{FE.Dec}(\cdot, c) : \text{dk}_f \mapsto \text{FE.Dec}(\text{dk}_f, c)$ be the univariate function associated to FE.Dec obtained by fixing c . We make the following assumptions:*

- For any parameter $\lambda \in \mathbb{N}$, pair of keys $(\text{msk}, \text{mpk}) \leftarrow \text{FE.Setup}(1^{\lambda_2})$ and function $f \in \mathcal{F}_{\text{FE}}$, $\text{FE.KDer}_{\text{msk}}(f) \rightarrow \text{dk}_f$ lies in the message space $\mathcal{M}_{\text{HE}}$ of HE;
- For any ciphertext $c \leftarrow \text{FE.Enc}_{\text{mpk}}(x)$, $x \in \mathcal{M}_{\text{FE}}$, $\text{FE.Dec}(\cdot, c)$ is in the class of functions $\mathcal{F}_{\text{HE}}$.

Under these assumptions, the functional decryption algorithm can be run homomorphically, that is, for any tuple of keys $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{HE.KGen}(1^{\lambda_1})$ and encryption $c \leftarrow \text{FE.Enc}_{\text{mpk}}(x)$,

$$\mathbb{P}\left[\text{HE.Eval}_{\text{evk}}\left(\text{FE.Dec}(\cdot, c), \text{HE.Enc}_{\text{pk}}(\text{dk}_f)\right) \notin \text{Im}\left(\text{HE.Enc}_{\text{pk}}\left(f(x)\right)\right)\right] \quad (2)$$

is negligible in λ_1 and λ_2 .

Proof. Let $(\lambda_1, \lambda_2) \in \mathbb{N}^2$ be arbitrary security parameters, for HE and FE respectively. Let $f \in \mathcal{F}_{\text{FE}}$ be an arbitrary function and $m \in \mathcal{M}_{\text{FE}}$ a random message. Run $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{HE.KGen}(1^{\lambda_1})$, $(\text{msk}, \text{mpk}) \leftarrow \text{FE.Setup}(1^{\lambda_2})$ and $c \leftarrow \text{FE.Enc}_{\text{mpk}}(x)$. For the sake of brevity and readability, we use the following notations: $c_{\text{dk}_f} := \text{HE.Enc}_{\text{pk}}(\text{dk}_f)$ and $g := \text{FE.Dec}(\cdot, c)$. The first member of [Equation \(2\)](#) becomes $\text{HE.Dec}_{\text{sk}}(\text{HE.Eval}_{\text{evk}}(g, c_{\text{dk}_f}))$, where $g \in \mathcal{F}_{\text{HE}}$ by assumption. The homomorphic properties of HE guarantee the following equality:

$$p_1 := \mathbb{P}\left[\text{HE.Dec}_{\text{sk}}(\text{HE.Eval}_{\text{evk}}(g, c_{\text{dk}_f})) \neq g(\text{HE.Dec}_{\text{sk}}(c_{\text{dk}_f}))\right] = \text{negl}(\lambda_1) \quad (3)$$

by correctness of HE. Besides,

$$p_2 := \mathbb{P}[g(\text{HE.Dec}_{\text{sk}}(c_{\text{dk}_f})) \neq g(\text{dk}_f)] = \text{negl}(\lambda_1) \quad (4)$$

because g is deterministic and HE is correct in the sense of PKE. Finally, as $g(\text{dk}_f) = \text{FE.Dec}(\cdot, c)(\text{dk}_f) = \text{FE.Dec}(\text{dk}_f, c)$ by definition, we have:

$$p_3 := \mathbb{P}[g(\text{dk}_f) \neq f(x)] = \text{negl}(\lambda_2) \quad (5)$$

because FE is correct. Denote p_0 the probability given in [Equation \(2\)](#). Then $p_0 = p_1 + p_2 + p_3 = 2 \cdot \text{negl}(\lambda_1) + \text{negl}(\lambda_2) = \text{negl}(\lambda_1, \lambda_2)$. $\square$

4.2 Representation of Functions

Theorem 1 and **Theorem 2** are valid in theory, but when it comes to practice, the condition $f \in \mathcal{M}_{\text{HE}}$ stated in **Theorem 1** can be hard to capture. In **section 2**, we highlight the importance of IPFE, a class of FE schemes that allows the evaluation of the inner product over encrypted data. In this scenario, the function $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$ can be represented by the vector $\mathbf{y}$, which is sent by the analyst to the curator. In this case, it is easy to choose $\mathbf{y}$ and HE such that $\mathbf{y} \in \mathcal{M}_{\text{HE}}$.

In the case of Quadratic Functional Encryption (QFE) [Gay16, BCFG17, Tom23], the function is a quadratic polynomial $P = \sum a_{ij} X^i Y^j$. As for IPFE, this function is entirely determined by a finite set of coefficients $\{a_{ij}\}$ that can be represented as a vector in $\mathcal{M}_{\text{HE}}$. Finally, we mention the case of SPADE [NAM24], an FE scheme in which the function is an equality test for a queried value v . This particular FE function is entirely determined by v and therefore has a straightforward representation as an element of $\mathcal{M}_{\text{HE}}$. In the general case, we define a function as follows:

Definition 4 (Representation of a Function). Let $f : \mathcal{X} \rightarrow \mathcal{Y}$ be an arbitrary function for sets $\mathcal{X}$ and $\mathcal{Y}$. Then f is entirely determined by the image $f(x) \in f(\mathcal{X}) \subseteq \mathcal{Y}$ of each element $x \in \mathcal{X}$. In practice, $f := \{(x, f(x)) \in \mathcal{X} \times f(\mathcal{X})\}$. Moreover, if f is linear and $\mathcal{X}$ is finitely generated, $f := \{(\mathbf{e}_i, f(\mathbf{e}_i))\}$, for all generators $\mathbf{e}_i$ of $\mathcal{X}$.

In the context of FE, $f := ((x_1, f(x_1)), \dots, (x_n, f(x_n)))$, where $x_i \in \mathcal{M}$ is an element of the FE message space and $f(x_i)$ its evaluation by $f \in f(\mathcal{M})$. With a proper encoding, each pair can be represented as a plaintext of HE hence allowing the practical encryption of f .

Example: Let $\mathcal{X} := \mathbf{Z}^n$, $\mathcal{Y} := \mathbf{Z}$ and $f_{\mathbf{y}} : \mathbf{Z}^n \rightarrow \mathbf{Z}; \mathbf{x} \mapsto \langle \mathbf{x}, \mathbf{y} \rangle$, for $\mathbf{y} \in \mathbf{Z}^n$. $f_{\mathbf{y}}$ is naturally determined by the set $\{(\mathbf{x}, \langle \mathbf{x}, \mathbf{y} \rangle)\}_{\mathbf{x} \in \mathbf{Z}^n}$. However, since $f_{\mathbf{y}}$ is a group morphism and $\mathbf{Z}^n$ is an additive group generated by the canonical basis $(\mathbf{e}_1, \dots, \mathbf{e}_n)$, it is sufficient to consider the representation $\{(\mathbf{e}_i, \langle \mathbf{e}_i, \mathbf{y} \rangle)\}_{i \in n}$, that is $\{(\mathbf{e}_i, y_i)\}$, that is $\mathbf{y} = (y_1, \dots, y_n)$, which is exactly what we do in practice.

5 Homomorphic Functional Encryption (HFE)

This section provides a formal definition of the encryption scheme obtained by evaluating the FE algorithms homomorphically.

Definition 5 (Homomorphic Functional Encryption). Let $\text{HE} := (\text{KGen}, \text{Enc}, \text{Eval}, \text{Dec})$ be an exact homomorphic encryption scheme and $\text{FE} := (\text{Setup}, \text{Enc}, \text{KDer}, \text{Dec})$ be a non-noisy functional encryption scheme for message space $\mathcal{M}$ and functionality space $\mathcal{F}$, such that **Theorem 1** is applicable. An HFE scheme for message space $\mathcal{M}$ and class of functions $\mathcal{F}$ consists of five PT algorithms HFE.Setup, HFE.Encap, HFE.Enc, HFE.KDer, HFE.Dec defined as follows:

- **HFE.Setup** (1^λ). The setup algorithm is a probabilistic algorithm. On input it takes a security parameter λ ; computes $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{HE.KGen}(1^\lambda)$ and outputs the secret key sk , the public key pk and the evaluation key evk ;
- **HFE.Encap** ($1^\mu, \text{pk}$). The encapsulation algorithm is a probabilistic algorithm that takes on input a security parameter μ and the public key pk . It generates the master secret and public keys $(\text{msk}, \text{mpk}) \leftarrow \text{FE.Setup}(1^\mu)$, encrypts $c_{\text{msk}} \leftarrow \text{HE.Enc}_{\text{pk}}(\text{msk})$ and outputs the master public key mpk and the homomorphically encrypted master secret key c_{msk} ;
- **HFE.Enc** (mpk, x). The encryption algorithm is a probabilistic algorithm. On the input of public key mpk and a message x in the message space $\mathcal{M}$, outputs a ciphertext $c \leftarrow \text{FE.Enc}_{\text{mpk}}(x)$;

- $\text{HFE.KDer}(c_{\text{msk}}, \text{evk}, f)$. The key derivation algorithm is a deterministic algorithm. On input it takes the homomorphically encrypted key c_{msk} , the evaluation key evk and a function $f \in \mathcal{F}$. It encrypts homomorphically the function f into $c_f \leftarrow \text{HE.Enc}_{\text{pk}}(f)$ and outputs a homomorphically encrypted decryption key $\text{HE.Eval}_{\text{evk}}(\text{FE.KDer}, (c_{\text{msk}}, c_f)) \rightarrow c_{\text{dk}_f}$;
- $\text{HFE.Dec}(\text{sk}, c_{\text{dk}_f}, c)$; The decryption algorithm is a deterministic algorithm. On input it takes the secret homomorphic key sk , a homomorphically encrypted decryption key c_{dk_f} and a ciphertext c ; computes $\text{HE.Dec}_{\text{sk}}(c_{\text{dk}_f}) \rightarrow \text{dk}_f$ and outputs a result $\text{FE.Dec}(\text{dk}_f, c) \rightarrow \text{res}$.

Correctness: For all parameters $\lambda, \mu \in \mathbb{N}$, $f \in \mathcal{F}$, $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{HFE.Setup}(1^\lambda)$, $(\text{mpk}, c_{\text{msk}}) \leftarrow \text{HFE.Encap}(1^\mu, \text{pk})$, $\text{HFE.KDer}(c_{\text{msk}}, \text{evk}, f) \rightarrow c_{\text{dk}_f}$ and $x \in \mathcal{M}$,

$$\mathbb{P}[\text{HFE.Dec}(\text{sk}, c_{\text{dk}_f}, \text{HFE.Enc}(\text{mpk}, x)) \neq f(x)] = \text{negl}(\lambda, \mu)$$

Remark 3 (Correctness). Homomorphic functional encryption is correct by construction: Let HFE be a homomorphic functional encryption scheme constructed after an HE scheme HE and an FE scheme FE. Let $\lambda, \mu \in \mathbb{N}$, $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{HFE.Setup}(1^\lambda)$, $(\text{mpk}, c_{\text{msk}}) \leftarrow \text{HFE.Encap}(1^\mu, \text{pk})$, $f \in \mathcal{F}$, $\text{HFE.KDer}(c_{\text{msk}}, \text{evk}, f) \rightarrow c_{\text{dk}_f}$ and $x \in \mathcal{M}$, $c \leftarrow \text{HFE.Enc}(\text{mpk}, x)$, then

$$\begin{aligned} & \mathbb{P}[\text{HFE.Dec}(\text{sk}, c_{\text{dk}_f}, c) \neq f(x)] \\ &= \mathbb{P}[\text{FE.Dec}(\text{HE.Dec}(\text{sk}, c_{\text{dk}_f}), c) \neq f(x)] \\ &\leq \mathbb{P}[\{\text{FE.Dec}(\text{dk}_f, c) \neq f(x)\} \cup \{\text{HE.Dec}(\text{sk}, c_{\text{dk}_f}) \neq \text{dk}_f\}] \\ &\leq \mathbb{P}[\text{FE.Dec}(\text{dk}_f, c) \neq f(x)] + \mathbb{P}[\text{HE.Dec}(\text{sk}, c_{\text{dk}_f}) \neq \text{dk}_f] \\ &\leq \text{negl}(\lambda) + \text{negl}(\mu) \\ &= \text{negl}(\lambda, \mu). \end{aligned}$$

5.1 Function Blindness

This section does not aim at providing a formal security proof, but rather to give a grasp of how HFE achieves the security level we claim. Its main asset being the *obfuscation* of FE circuits using HE, we give an intuition of how HFE ensures function blindness.

Definition 6 (Function Blindness). An FE scheme is said to be function-blind if no PPT adversary, given two ciphertexts $c_1 \in \text{Im}(\text{Enc}(\text{pk}, x_1))$ and $c_2 \in \text{Im}(\text{Enc}(\text{pk}, x_2))$, decryption key requests req_{f_1} , req_{f_2} and a key generation algorithm $\text{KDer} : \text{req}_f \mapsto \text{dk}_f$, can't distinguish between req_{f_1} and req_{f_2} or c_1 and c_2 with non-negligible probability.

We propose a systematic method to achieve function blindness. This technique relies on the homomorphic encryption of f , prior to derivation. We present [Algorithm 1](#), which involves two parties: **U** (user) and **KC** (key curator). In this scenario, **U** sends a request to **KC** to obtain a decryption key for a function f , while ensuring that f remains hidden from **KC**.

Algorithm 1 Function-Blind Key Derivation

- Input:** HE keys $(\text{sk}, \text{pk}, \text{evk})$, the master secret key msk and a function $f \in \mathcal{F}$
Output: the decryption key dk_f
- 1: $\mathbf{U}$ encrypts $c_f \leftarrow \text{HE.Enc}_{\text{pk}}(f)$ and sends it to $\mathbf{KC}$
 - 2: $\mathbf{KC}$ encrypts $c_{\text{msk}} \leftarrow \text{HE.Enc}_{\text{pk}}(\text{msk})$
 - 3: $\mathbf{KC}$ runs $c \leftarrow \text{HE.Eval}_{\text{evk}}(\text{FE.KDer}(c_{\text{msk}}, \cdot), c_f)$ and sends it to $\mathbf{U}$
 - 4: $\mathbf{U}$ decrypts $\text{dk}_f \leftarrow \text{HE.Dec}_{\text{sk}}(c)$
-

Correctness: The output dk_f of [Algorithm 1](#) is indeed the decryption key for function f .

Proof. This is straightforward, as it relies on the correctness of the underlying HE and FE schemes. Therefore, we provide a short and intuitive proof.

$$\begin{aligned}
 \text{HE.Dec}_{\text{sk}}(c) &= \text{HE.Dec}_{\text{sk}}\left(\text{HE.Eval}_{\text{evk}}(\text{FE.KDer}_{\text{msk}}(\cdot), c_f)\right) \\
 &= \text{FE.KDer}_{\text{msk}}(\text{HE.Dec}_{\text{sk}}(c_f)) \\
 &= \text{FE.KDer}_{\text{msk}}(\text{HE.Dec}_{\text{sk}}(\text{HE.Enc}_{\text{pk}}(f))) \\
 &= \text{FE.KDer}_{\text{msk}}(f) = \text{dk}_f
 \end{aligned}$$

□

Function-blindness: Assuming that $\mathbf{KC}$ knows the functional secret key msk , a homomorphic encryption of the functional secret key c_{msk} and the public homomorphic keys (pk, evk) , [Algorithm 1](#) is function-blind. In other words, $\mathbf{KC}$ learns only negligible information about f during the key generation.

Proof (Intuition). We assume that $\mathbf{KC}$ has access to (pk, evk) and c_{msk} , and first assume that $\mathbf{KC}$ is a passive eavesdropper. The HE scheme HE is IND-HE-CPA secure, as defined in [Definition 8](#). This means that without the knowledge of sk , $\mathbf{KC}$ can not distinguish between c_f and a random element of $\mathcal{C}$ with a non-negligible advantage. Moreover, for any function (or circuit) g supported by HE, the homomorphic evaluation of g on a finite number of ciphertexts gives a negligible advantage to $\mathbf{KC}$ to distinguish the said ciphertexts. Since FE.KDer is, as stated in [subsection 4.1](#), a function supported by HE, the advantage of $\mathbf{KC}$ as a passive eavesdropper is negligible.

However, the problem with function-blindness in a public key setup is that $\mathbf{KC}$ can evaluate the function f in finitely many points, hence gathering non-negligible information on f . If we consider $\mathbf{KC}$ as an active eavesdropper, we need to guarantee that it can not conduct this type of attack with c_f only. We assume that $\mathbf{KC}$ can adaptively choose a finite number of elements $x_1, \dots, x_n$ in the plaintext space of FE and get their encryption $\mu_1, \dots, \mu_n$ under mpk . Following, $\mathbf{KC}$ can re-encrypt these ciphertexts into homomorphic ciphertexts, that is $c_i \leftarrow \text{HE.Enc}_{\text{pk}}(\mu_i)$, and run the functional decryption algorithm, as described in [subsection 4.1](#). More formally, given c a homomorphic encryption of dk_f and c_i a homomorphic ciphertext for which it knows the underlying plaintext, $\mathbf{KC}$ can compute:

$$\text{HE.Eval}_{\text{evk}}\left(\text{FE.Dec}(c, c_i)\right) = c_{f(x_i)} \in \text{Im}\left(\text{HE.Enc}(f(x_i))\right) \quad (6)$$

by correctness of HFE. The major result here is that $\mathbf{KC}$ performs only computations on homomorphic ciphertexts. The attack aforementioned only returns an encryption of the result $f(x_i)$ and therefore leaks no information about the function f as HE is IND-HE-CPA. □

To conclude, the HFE scheme defined in [Definition 5](#) is function-blind.

6 Semantic Security

This section provides a detailed and formal definition of the semantic security required for an HFE scheme. In addition, we formally demonstrate the semantic security of HFE with respect to our definition. This definition relies on the traditional notion of IND-CPA security for HE and FE, since it is the most commonly used paradigm. We provide [Theorem 3](#), which ensures the theoretical security of HFE, given that HE and FE are IND-CPA secure.

6.1 Indistinguishability-based Security

While the security of PKE is well-studied, it remains challenging to define a universal security definition for HE and FE. The reason is two-fold: firstly, the capacity to make computations on the ciphertexts clearly impacts the notion of *indistinguishability*. This is particularly true in FE, where the decryption algorithm acts as a *trapdoor*, hence challenging traditional security definitions. Secondly, the application case and the considered system model significantly impact the security expectations. If these concerns are usually delegated to the security of the protocol, the large panel of use-cases offered by both HE and FE requires the user to ask what oracle can reasonably be provided to an attacker, even at the scheme's semantic security level. A good example of this problem is the recent introduction by Li and Micciancio [\[LM21\]](#) of IND-CPA^D. In short, it consists of providing the attacker a conditional access to the decryption oracle. This seems like a reasonable assumption, since HE can provide the decryption of particular results to a non-fully-trusted entity. However, there is yet no consensus on this level of security [\[CCP⁺24, CSBB24\]](#), which is highly influenced by the considered use case.

FE Indistinguishability under Chosen Plaintext Attack: This security notion is usually formalized through a security game. The game starts with a challenger flipping a truly random coin $\beta \in \{0, 1\}$, generating the master secret and public keys, and sending the master public key to the adversary $\mathcal{A}$. The latter subsequently requests functional decryption keys for functions of its choice. At some point, $\mathcal{A}$ outputs two messages x_0, x_1 to the challenger, which replies with c_β . The scheme is IND-FE-CPA secure if $\mathcal{A}$ cannot guess the value of β with probability greater than $1/2 + \text{negl}$. More formally, the security game is described below:

Definition 7 (IND-CPA Security for FE (IND-FE-CPA)). We say that an FE scheme FE is IND-FE-CPA secure, if for all PPT adversary $\mathcal{A}$, it holds that the advantage of $\mathcal{A}$:

$$\text{Adv}_{\text{FE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda) = \left| \Pr \left[\text{Exp}_{\text{FE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda) \rightarrow \text{true} \right] - \frac{1}{2} \right|$$

is negligible in λ , where the experiment is defined as follows in [Figure 1](#).

$$\begin{array}{l} \text{Exp}_{\text{FE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda) : \\ \quad \beta \leftarrow \{0, 1\}; L_K, L_M \leftarrow \emptyset \\ \quad (\text{msk}, \text{mpk}) \leftarrow \text{Setup}(1^\lambda) \\ \quad \beta' \leftarrow \mathcal{A}(1^\lambda, \text{mpk} : \mathcal{O}_{\text{ind-FE}}) \\ \quad \text{Return } (\beta = \beta') \end{array}$$

The game records the queries made by $\mathcal{A}$ through a list L_M storing encryption queries and a list L_F storing functional queries. Each request is made under the condition that $f_0(x_0) = f_1(x_1)$, for any $(f_0, f_1) \in L_F$ and $(x_0, x_1) \in L_M$. This condition is mandatory, as violating it gives $\mathcal{A}$ an overwhelming advantage to distinguish $f_0(x_0)$ from $f_1(x_1)$.

HE Indistinguishability under Chosen Plaintext Attack: The security notion of IND-HE-CPA security is usually formalized through a security game. The game starts

Oracles $\mathcal{O}_{\text{ind-FE}}$

```

ENCRYPT ( $\text{mpk}, x_0, x_1$ )
  If  $x_0, x_1 \notin \mathcal{M}$ :
    return  $\perp$ 
  If  $\exists (f_0, f_1) \in L_F ; f_0(x_0) \neq f_1(x_1)$ :
    return  $\perp$ 
   $c_\beta \leftarrow \text{Enc}(\text{mpk}, x_\beta)$ 
   $L_M \leftarrow L_M \cup \{(x_0, x_1, c_\beta)\}$ 
  return  $c_\beta$ 
KEYDER ( $f_0, f_1$ )
  If  $f_0, f_1 \notin \mathcal{F}$ :
    return  $\perp$ 
  If  $\exists (x_0, x_1) \in L_M ; f_0(x_0) \neq f_1(x_1)$ :
    return  $\perp$ 
   $\text{dk}_{f_\beta} \leftarrow \text{KDer}(\text{msk}, f_\beta, D)$ 
   $L_F \leftarrow L_F \cup \{(f_0, f_1, \text{dk}_{f_\beta})\}$ 
  return  $\text{dk}_{f_\beta}$ 

```

Figure 1: Oracles in the IND-FE-CPA game for an FE scheme FE and PPT adversary $\mathcal{A}$: the encryption oracle ENCRYPT and the key derivation oracle KEYDER.

with a challenger $\mathcal{C}$ flipping a truly random coin $\beta \in \{0, 1\}$, generating the public, secret and evaluation keys, and sending the public key to the adversary $\mathcal{A}$. The latter has access to an encryption oracle, that on the input of two messages x_0, x_1 outputs the ciphertext c_β . Subsequently, $\mathcal{A}$ requests evaluations for functions of its choice. At some point, $\mathcal{A}$ outputs a guess β' on the value of β . The scheme is IND-FE-CPA secure if $\mathcal{A}$ cannot guess the value of β with probability greater than $1/2 + \text{negl}$. More formally, the security game is described below:

Definition 8 (IND-CPA Security for HE (IND-FE-CPA)). Let $\text{HE} = (\text{KGen}, \text{Enc}, \text{Eval}, \text{Dec})$ be an HE scheme for message space $\mathcal{M}$ and functionality space $\mathcal{F} : \mathcal{M}^n \rightarrow \mathcal{M}$. We say that HE is IND-FE-CPA secure if for all PPT adversary $\mathcal{A}$, it holds that the advantage of $\mathcal{A}$ given by $\text{Adv}_{\text{HE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda) = \left| \Pr \left[\text{Exp}_{\text{HE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda) \rightarrow \text{true} \right] - \frac{1}{2} \right|$ is negligible in λ , where the experiment is defined as follows:

```

 $\text{Exp}_{\text{HE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda)$ :
   $\beta \leftarrow \{0, 1\}; L \leftarrow \emptyset$ 
   $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{KGen}(1^\lambda)$ 
   $\beta' \leftarrow \mathcal{A}(1^\lambda, \text{pk}, \text{evk}, : \mathcal{O}_{\text{ind-FE}})$ 
  return  $(\beta = \beta')$ 

```

The game records the queries made by $\mathcal{A}$ through a list L storing all the ciphertexts it received. This includes encryption of a plain message sent by $\mathcal{A}$ as well as ciphertexts resulting from the evaluation of a function f .

6.2 Security of HFE

Similarly to the security of FE defined in Definition 7, we formalize IND-CPA through a security game. The challenger picks randomly $\beta \in \{0, 1\}$, generates the keys, encapsulates the master secret key, and sends the public parameters to $\mathcal{A}$; that is the master public key mpk , the evaluation key evk and the encapsulation c_{msk} . The adversary subsequently requests functional decryption keys for functions of its choice. $\mathcal{A}$ eventually sends two

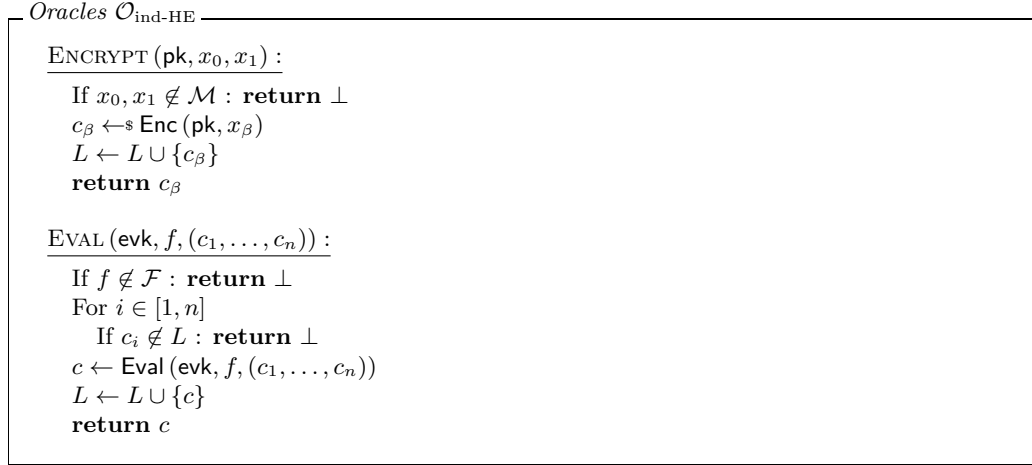


Figure 2: Oracles in the IND-HE-CPA game for an HE scheme HE and PPT adversary $\mathcal{A}$: the encryption oracle ENCRYPT and the evaluation oracle EVAL.

messages x_0, x_1 to the challenger, which replies with c_β . The scheme is IND-CPA secure if $\mathcal{A}$ cannot guess the value of β with probability greater than $1/2 + \text{negl}$. More formally, the security game is described below:

Definition 9 (IND-CPA Security for HFE (IND-HFE-CPA)). Let HFE be an HFE scheme for message space $\mathcal{M}$ and functionality space $\mathcal{F}$. We say that HFE is IND-HFE-CPA if for all PPT adversary $\mathcal{A}$, it holds that the advantage of $\mathcal{A}$:

$$\text{Adv}_{\text{HFE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda, \mu) = \left| \mathbb{P} \left[\text{Exp}_{\text{HFE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda, \mu) \rightarrow \text{true} \right] - \frac{1}{2} \right|$$

is negligible in λ and μ , where the experiment is defined as follows:

$$\begin{aligned} & \text{Exp}_{\text{HFE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda, \mu): \\ & \quad \beta \leftarrow \{0, 1\}; L_M \leftarrow \emptyset; L_F \leftarrow \emptyset \\ & \quad (\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{Setup}(1^\lambda) \\ & \quad (\text{mpk}, c_{\text{msk}}) \leftarrow \text{Encap}(1^\mu, \text{pk}) \\ & \quad \beta' \leftarrow \mathcal{A}(1^\lambda, 1^\mu, \text{mpk}, \text{pk}, \text{evk} : \mathcal{O}_{\text{ind-HFE}}) \\ & \quad \text{return } (\beta == \beta') \end{aligned}$$

As for FE, the game keeps record of the queries made by $\mathcal{A}$ through a list L_M storing all the encryption queries and a list L_F storing each functional queries, represented as $(f_0, f_1, c_{\text{dk}_{f_\beta}})$, where f_0, f_1 are the queried functions and $c_{\text{dk}_{f_\beta}}$ is the output of the HFE key derivation algorithm. Each request is made under the condition that $f_0(x_0) = f_1(x_1)$, for any $(f_0, f_1, c_{\text{dk}_{f_\beta}}) \in L_F$ and $(x_0, x_1, c_\beta) \in L_M$. This condition is mandatory, as violating it gives $\mathcal{A}$ an overwhelming advantage to distinguish $f_0(x_0)$ from $f_1(x_1)$. Note that the EVAL algorithm works slightly differently than in a classical HE scheme. Even though the output of the evaluation for a function g is a valid homomorphic ciphertext, it is not necessarily the encryption of a valid functional decryption key for, *e.g.* dk_g . However, $\mathcal{A}$ can use this oracle to try to get valuable information for $\text{dk}_{f_1}, \dots, \text{dk}_{f_n}$. Finally, a decryption oracle DECRYPT outputs the functional decryption key $f(\text{dk}_{f_\beta})$ as in a traditional FE scheme.

Theorem 3. Let HFE be an HFE scheme built from an HE scheme HE and an FE scheme FE. If HE and FE are IND-HE-CPA and IND-FE-CPA secure respectively, then HFE is IND-HFE-CPA secure.

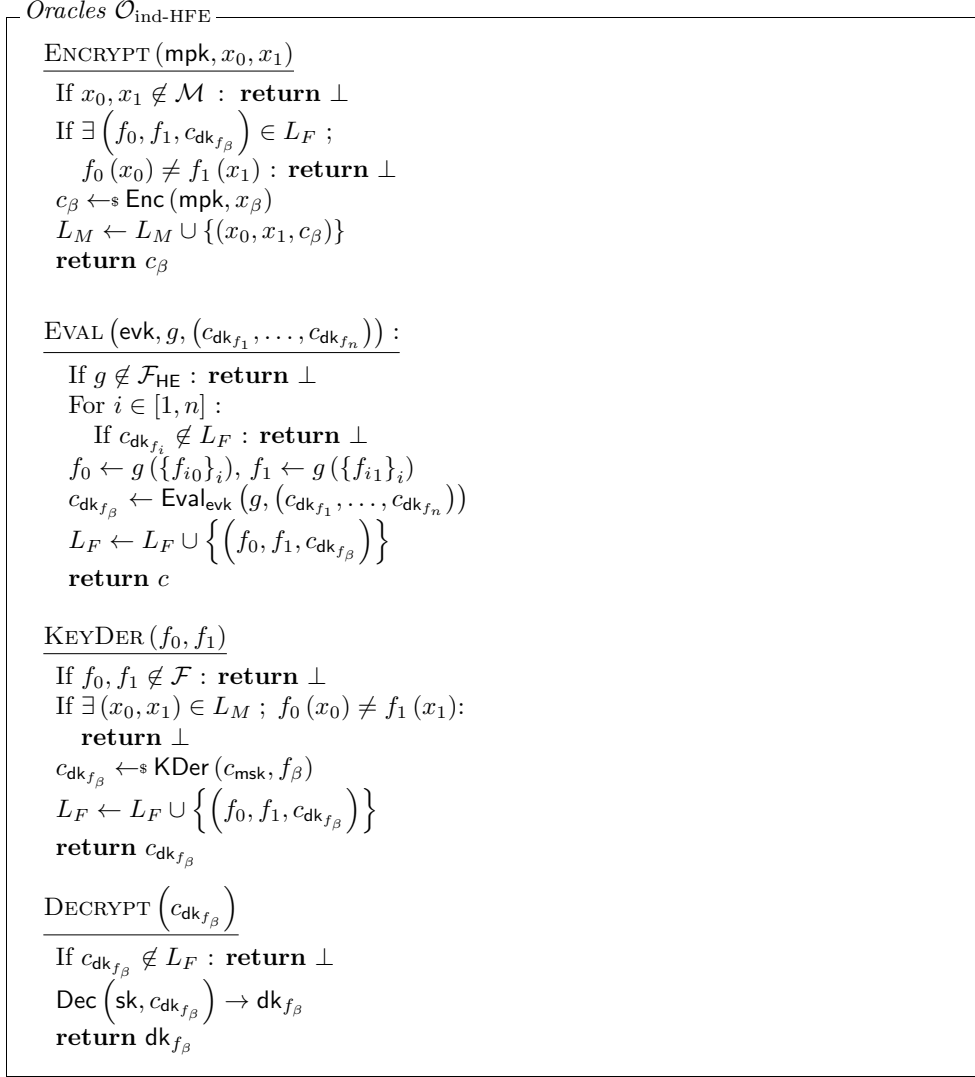


Figure 3: Oracles in the IND-HFE-CPA game for a HFE scheme HFE and PPT adversary $\mathcal{A}$: The encryption oracle ENCRYPT, the key derivation oracle KEYDER, the evaluation oracle EVAL and a decryption oracle DECRYPT.

Proof. We prove [Theorem 3](#) through a sequence of games, starting with the genuine HFE game and ending with a game where the advantage of $\mathcal{A}$ is negligible. We denote ϵ_i the advantage of $\mathcal{A}$ for game i and $\epsilon(\lambda)$ (*resp.* $\epsilon(\mu)$) its advantage for the HE (*resp.* FE) game.

Game 0: This is the initial genuine HFE game defined in [Definition 9](#). The challenger initializes the game by flipping at random a bit $\beta \leftarrow \{0, 1\}$. Then, the key generation goes as follows: on the one hand, a PPT adversary $\mathcal{A}$ samples the homomorphic keys $(\text{pk}, \text{sk}, \text{evk}) \leftarrow \text{Setup}(1^\lambda)$ and forwards pk to $\mathcal{C}$ who runs the encapsulation $(\text{msk}, c_{\text{msk}}) \leftarrow \text{Encap}(1^\mu, \text{pk})$. $\mathcal{A}$ has access to the oracles ENCRYPT, EVAL, KEYDER and EVAL. Eventually, $\mathcal{A}$ outputs a guess β' on the value of β with an advantage ϵ_0 .

Game 1: This game is similar to the previous one, except that the oracle DECRYPT is replaced by an oracle $\text{DECRYPT}_{\mathbf{G}_1}$ defined as follows:

$$\text{DECRYPT}_{\mathbf{G}_1} \left(c_{\text{dk}_{f_\beta}} \right) :$$

If $c_{\text{dk}_{f_\beta}} \notin L_F$: **return** $\perp$
 $\text{dk} \leftarrow \text{Im}(\text{FE.KDer}_{\text{msk}})$
return dk

In short, when $\mathcal{A}$ calls the DECRYPT oracle, $\mathcal{C}$ runs the $\text{DECRYPT}_{\mathbf{G}_1}$ oracle instead, which returns a random functional decryption key dk as plain, as in the classical FE game. We prove thereunder that a PPT adversary cannot distinguish this game from the previous one (with a non-negligible advantage).

Claim 1. $|\epsilon_0 - \epsilon_1| \leq \epsilon(\mu)$, where $\epsilon(\mu)$ is the advantage of an efficient adversary that breaks the FE game.

To prove this claim, we consider a PPT adversary $\mathcal{B}$ that breaks the IND-FE-CPA game with an advantage $\epsilon(\mu)$. $\mathcal{A}$ constructs an oracle $\text{KEYDER}_{\mathbf{G}_1}$ defined as follows:

$$\text{KEYDER}_{\mathbf{G}_1}(f_0, f_1)$$

$c_{\text{dk}_{f_\beta}} \leftarrow \text{KEYDER}(f_0, f_1)$
 $\text{dk}_{f_\beta} \leftarrow \text{DECRYPT}(\text{sk}, c_{\text{dk}_{f_\beta}})$
return dk_{f_β}

Note that the oracle $\text{KEYDER}_{\mathbf{G}_1}$ is well-defined, as it only calls oracle accessible to $\mathcal{A}$. One also notices that $\text{KEYDER}_{\mathbf{G}_1}$ corresponds to the classical key derivation oracle KEYDER defined in Figure 1. Therefore, $\mathcal{A}$ provides to $\mathcal{B}$ the master public key mpk and access to the oracles ENCRYPT and KEYDER . $\mathcal{B}$ plays the IND-FE-CPA game using these data and outputs to $\mathcal{A}$ a guess β'' on the value β with an advantage $\epsilon(\mu)$, which concludes.

Game 2: This game proceeds as the previous one, except that we replace the KEYDER oracle with a new oracle $\text{KEYDER}_{\mathbf{G}_2}$, defined as follows:

$$\text{KEYDER}_{\mathbf{G}_2}(f_0, f_1)$$

If $f_0, f_1 \notin \mathcal{F}$: **return** $\perp$
 $c \leftarrow \text{Im}(\text{HE.Enc})$
 $L_F \leftarrow L_F \cup \{(f_0, f_1, c)\}$
return c

In this game, when $\mathcal{A}$ calls the KEYDER oracle, $\mathcal{C}$ runs the output of $\text{KEYDER}_{\mathbf{G}_2}$ instead, and hence substitutes a random element of the ciphertext space to the encryption of f_β . We prove that a PPT adversary cannot distinguish this game from the previous one (with a non-negligible advantage).

Claim 2. $|\epsilon_1 - \epsilon_2| \leq \epsilon(\lambda)$, where $\epsilon(\mu)$ is the advantage of an efficient adversary that breaks the HE game.

To prove this claim, we reduce Game 2 to the IND-FE-CPA game defined in Definition 8. We introduce an oracle $\text{ENCRYPT}_{\mathbf{G}_2}$ defined as follows:

$$\text{ENCRYPT}_{\mathbf{G}_2}(x_0, x_1)$$

$c_\beta \leftarrow \text{KEYDER}(x_0, x_1)$
return c_β

As before, we consider a second PPT adversary $\mathcal{B}$ that has an advantage $\epsilon(\lambda)$ in the IND-HE-CPA game. The oracle $\text{ENCRYPT}_{\mathbf{G}_2}$ is well-defined, as it only calls oracle accessible to $\mathcal{A}$, and corresponds to the classical homomorphic encryption oracle ENCRYPT defined in Figure 2. During the IND-HFE-CPA game, $\mathcal{A}$ can forward (pk, evk) to $\mathcal{B}$. With access to both oracles EVAL and $\text{ENCRYPT}_{\mathbf{G}_2}$, $\mathcal{B}$ outputs a guess β'' on the value β with an advantage $\epsilon(\lambda)$, which concludes.

Game 3: This game proceeds as the previous one, except that we replace the ENCRYPT oracle with a new oracle $\text{ENCRYPT}_{\mathbf{G}_3}$, defined as follows:

```

ENCRYPTG3(mpk, x0, x1)
  If x0, x1 ∉ M : return ⊥
  c ←$ lm(Enc)
  return c

```

In this game, when $\mathcal{A}$ calls the ENCRYPT oracle, $\mathcal{C}$ runs the output of ENCRYPT_{G3} instead, and hence substitutes a random element of the ciphertext space to the encryption of x_β . We prove that a PPT adversary cannot distinguish this game from the previous one (with a non-negligible advantage).

Claim 3. $|\epsilon_3 - \epsilon_2| \leq \epsilon'(\mu)$, where $\epsilon'(\mu)$ is the advantage of an efficient adversary that breaks the FE game in the sense of PKE.

The key argument to prove this claim is that the functional encryption c_β is not used in any oracle but the encryption, as KEYDER has been replaced by random oracles in the previous games. Hence, distinguishing the output c_β of the algorithm ENCRYPT from a random element $c \leftarrow_{\$} \text{lm}(\text{FE.Enc})$ is reduced to a traditional IND-CPA game in the sense of PKE. The advantage $\epsilon'(\mu)$ of an efficient adversary $\mathcal{A}$ in such a game is negligible.

Conclusion: Every oracle of Game 3 is independent from β , hence $\mathcal{A}$ has a nil advantage in Game 3. The overall advantage ϵ_{HFE} of $\mathcal{A}$ in the HFE game defined in Definition 9 is $\epsilon_{\text{HFE}} \leq \epsilon(\mu) + \epsilon(\lambda) + \epsilon(\mu')$. As a finite sum of negligible elements, ϵ_{HFE} is negligible, which concludes. $\square$

7 Real-World Deployment Scenario for HFE

In this section, we present a three-party toy protocol, demonstrating how our HFE scheme can be utilized in real-world applications. Consider a scenario in which a data analyst **DA** wants to perform selective computations on sensitive data x held by a user **U**. To do so, they rely on a third-party **KC**, or key curator, that should not know either x or the computations performed by **DA**.

Overview of the protocol: **DA** generates the homomorphic keys $(\text{sk}, \text{pk}, \text{evk})$ on one side, and the data owner generates the functional keys (msk, mpk) on the other side. The latter encrypts msk into a ciphertext c_{msk} with the analyst's homomorphic key pk and encrypts x into a ciphertext c under his functional public key mpk . The analyst, who has access to c , wants to calculate $f(x)$. She first encrypts f under pk into c_f and sends a request to **KC** to obtain (an encryption of) the functional decryption key c_{dk_f} . The curator, given the encrypted function c_f , the encrypted master secret key c_{msk} , and the evaluation key evk , runs homomorphically the functional key generation algorithm to output the encrypted decryption key c_{dk_f} . It then sends it back to the analyst who uses the homomorphic secret key sk to retrieve dk_f and finally $f(x)$.

System model: The model we consider consists of three entities: (i) Key Curator (**KC**), (ii) User (**U**) and (iii) Data Analyst (**DA**).

- **Key Curator (KC):** The curator is an authority responsible for the generation of decryption keys. Upon request of **DA** for a function f , **KC** is responsible for the generation of the corresponding decryption key dk_f ;
- **User (U):** The user holds the plain data x . He sets the system up by generating the functional parameters $(\text{msk}, \text{mpk}) \leftarrow_{\$} \text{FE.Setup}(1^\lambda)$ and sending $c_{\text{msk}} \leftarrow_{\$} \text{HE.Enc}_{\text{pk}}(\text{msk})$ to **KC**, where pk is the public key of **DA** for the HE scheme.
- **Data Analyst (DA):** The data analyst is an external party who wishes to learn the evaluation of a certain function f over data x provided by the user. With exclusive access to this as a ciphertext, **DA** sends a request for f to **KC**, which provides the corresponding decryption key dk_f , through which the result of the computation $f(x)$ can be obtained.

Threat model: Concerning the *semantic* security (IND-CPA) and function-blindness, we rely on the analyses we provide in other sections, namely [Theorem 3](#) and [Section 5.1](#) respectively. Concerning the security of the protocol, we trust the user **U** with the information he sends, and assume that he provides genuine data, performs the encapsulation using the right homomorphic key pk , and encrypts the data using the public key mpk . We do *not* require the key curator to be fully trusted. However, we assume that **KC** and **DA** *do not collude*, as the knowledge of both c_{msk} and sk is sufficient to retrieve msk and hence decrypt every ciphertext produced by **U**. This last assumption is mandatory in most FE construction relying on a curator, trusted or not. Indeed, the curator can generate a decryption key, and the data analyst can obtain valuable information from it. When implemented in practice, one assumes that **KC** will generate a reasonable amount of decryption keys – usually one. However, if it provides a larger number of decryption keys, **DA** can bypass the limited information she is originally supposed to get.

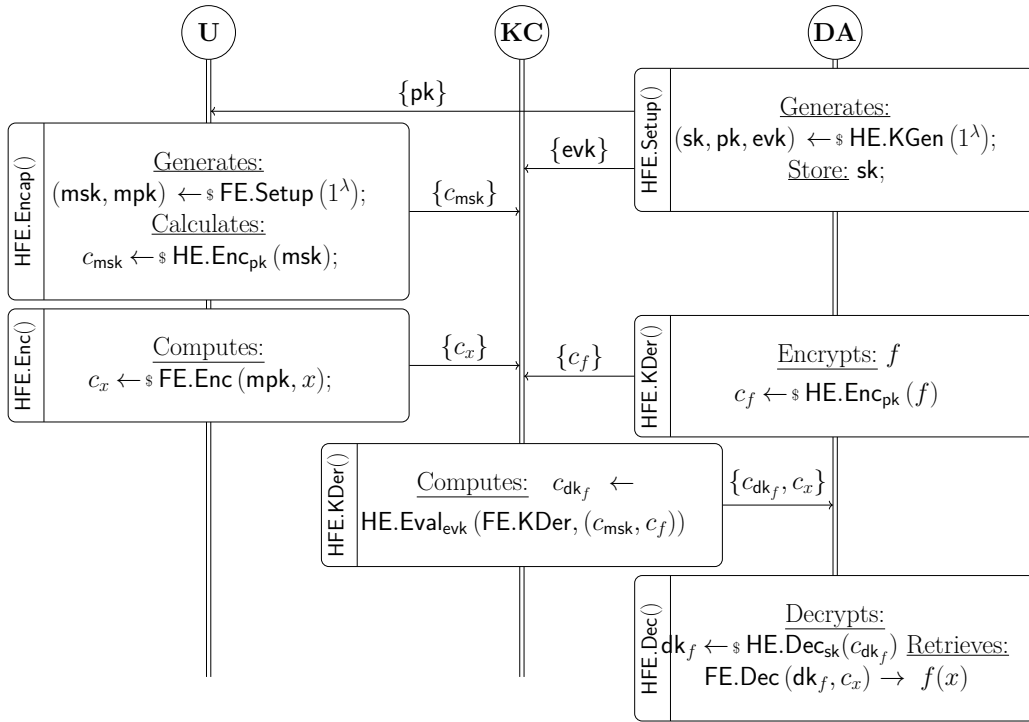


Figure 4: Protocol Flow Example.

In our protocol, we consider two arbitrary schemes: HE and FE for a space of the function $\mathcal{F}$. To give a more global impact to our article, we do not specify any scheme in particular, and assume that FE.KDer is a valid input of HE.Eval , and that we can express msk and $f \in \mathcal{F}$ as elements of the plaintext space of HE.

As shown in [Fig. 4](#), the setup consists of providing **KC** with all the information it requires to run the subsequent key derivation algorithm. On one side, **DA** generates the homomorphic parameters sk, pk , and evk . She stores the secret key sk for further decryption, sends pk to **U** and evk to **KC**. On the other side, the user generates the functional parameters msk, mpk . He encrypts homomorphically the master secret key into $c_{\text{msk}} \leftarrow \$ \text{HE.Enc}_{\text{pk}}(\text{msk})$ and sends it to **KC**.

After the system is set up, **U** can encrypt his data x . Using the public key mpk previously generated, he produces an FE ciphertext c_x and sends it to an external entity that can store it and forward it to **DA**. For the sake of simplicity, we consider that **KC** itself stores the ciphertexts, as it does not impact the practicability of our protocol.

The main component of HFE is the generation of the decryption key. Having an encryption c_{msk} of the master secret key and an encryption c_f of the function f , **KC** is capable of generating an encryption of the functional decryption key. To do that, it runs homomorphically (through the Eval algorithm) the decryption key generation, that is $\text{HE.Eval}_{\text{evk}}(\text{FE.KDer}, (c_{\text{msk}}, c_f))$. As explained in [Remark 1](#), this algorithm outputs a ciphertext c that lies in the image $\text{Im}(\text{HE.Enc}_{\text{pk}}(\text{dk}_f))$, that is the subset of $\mathcal{C}$ of the correct homomorphic encryptions of dk_f .

Using the secret key **sk**, **DA** can successfully retrieve dk_f . Following, **DA** runs the functional decryption algorithm as usual on the ciphertext c_x and finally gets the desired result $f(x)$.

Bound on the decryption key generation: As mentioned in the threat model, the security of FE always relies on a bound on the number of generated decryption keys. Regardless of the level of trust in it, **KC** has to keep records of the decryption keys it provides to **DA** and, in our model, of the decryption keys she could obtain with the evaluation algorithm. The *good practices* for the key generation can not be systematized, as the leakage produced by a decryption key depends on the FE scheme used.

8 Experiments

To evaluate the practicality of our HFE scheme, we implemented a proof-of-concept based on our real-world scenario shown in [Fig. 4](#). We chose the IPFE scheme of Mera *et al.* [\[MKMS22b\]](#) and the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [\[BGV14\]](#) as the underlying FE and HE schemes for HFE. Our implementation uses the *Python* programming language. For testing and benchmarking, we followed best practices by using the *pytest* and *pytest-benchmark* libraries. Additionally, we measured memory usage with the *memory-profiler* library. Our experiments were conducted in a single-threaded environment on a PC with an Intel Core i7-8700 CPU @ 3.20GHz and 64GB of memory.

To ensure a solid comparison, we present benchmark results of RLWE-based IPFE [\[MKMS22b\]](#) and HFE. The results include execution time, memory usage of various components, and storage costs associated with the multiple elements used during execution. Our comparison demonstrates that constructing a functional encryption scheme without relying on trusted entities is not only feasible but also comes with a reasonable overhead. We hope this work encourages further research and development toward trustless FE schemes.

Open Science: To support open science and reproducible research, we made our implementation source code available online².

8.1 Benchmarks

To benchmark RLWE-based IPFE and HFE, we follow the notation and algorithms in [Section 7](#), using a modular approach for implementing schemes, where $\text{FE} = \{\text{Setup}, \text{KDer}, \text{Enc}, \text{Dec}\}$ and $\text{HE} = \{\text{KGen}, \text{Enc}, \text{Eval}, \text{Dec}\}$. For simplicity, we denote the homomorphic evaluation process as KDerEval . As shown in [Table 1](#), we define three parameter sets—Params = $\{\mathbf{S}, \mathbf{M}, \mathbf{L}\}$ —to evaluate performance across different system scales. The small set (**S**) uses input vectors m and functions f of length 50 with a polynomial ring degree of $\log_2 N = 5$. The medium set (**M**) extends this to a length of 500 with $\log_2 N = 6$, while the large set (**L**) increases it to 1000 with $\log_2 N = 7$. Additionally, [Table 1](#) provides the ciphertext modulus, plaintext modulus, and bounds on the input and function vectors, denoted as $\log_2 q, \log_2 p, B_x, B_y$.

²<https://github.com/hosseinaabinf/HFE>

Table 1: Parameters for Test and Benchmark

Params	Vector Size	$\log_2 N$	$\log_2 q$	$\log_2 p$	B_x	B_y
Small	50	5	64	16	16	8
Medium	500	6	86	17	32	16
Large	1000	7	128	18	64	32

Running time and memory usage: As shown in Table 2, the running time of IPFE functions increases with the change of parameters. For the smallest benchmark (**S**), where vectors have a length of 50, the **Setup** phase takes $22.2ms$, while **Enc**, **KDer**, and **Dec** take $24.3ms$, $4.6ms$, and $5ms$, respectively. For the most significant benchmark (**L**), with vectors of length 1000, **Setup** extends to $2,988ms$, and **Enc**, **KDer**, and **Dec** require $3,098ms$, $243.7ms$, and $231.6ms$, respectively. However, as shown in Table 2, the memory usage of IPFE functions does not increase gradually. Even with larger vector lengths and input bounds, memory usage remains stable, averaging around $65MB$.

Similarly, as presented in Table 3, the running time of HFE functions increases as the parameters change. As expected, the execution times of the underlying FE functions **Setup**, **Enc**, **Dec** within HFE remain nearly identical to RLWE-based IPFE, with minor variations due to benchmarking conditions that are negligible. Since HFE employs BGV for homomorphic evaluation, its running time is also reported. For the smallest benchmark (**S**), with vector length 50, **HE.KGen** takes $21.3ms$, while **HE.Enc** and **HE.Dec** take $47.8ms$ and $0.4ms$, respectively. In the largest benchmark (**L**), with vector length 1000, **HE.KGen** extends to $258ms$, and **HE.Enc** and **HE.Dec** require $6,192ms$ and $3.1ms$, respectively. Most importantly, in HFE, the running time of **KDerEval** increases significantly, from $1054.7ms$ in benchmark (**S**) to $335,695.6ms$ in benchmark (**L**). However, as shown in Table 3, the memory usage of HFE functions does not increase gradually. Even with larger vector lengths and input bounds, memory usage remains stable, averaging around $84MB$.

An important remark is that, even in the largest use case, **KDerEval** requires around 5.6 minutes to generate a functional decryption key. *However, this operation only needs to be performed once per distinct function and can be carried out during the offline phase.* In this phase, the data analyst requests the desired function, obtains the corresponding keys, and can then use them to compute and decrypt results from stored ciphertexts an *unlimited* number of times.

Table 2: Running Time and Memory Costs of RLWE-based IPFE

Bench	Params	Time (ms)	Memory (MB)
Setup	S	22.2	55.99
	M	499.2	59.31
	L	2988	67.62
Enc	S	24.3	57.41
	M	521.1	61.56
	L	3098	76.99
KDer	S	4.6	67.51
	M	69.6	67.85
	L	243.7	70.69
Dec	S	5	67.06
	M	67.8	68.46
	L	231.6	76.50

Storage costs: As shown in Table 4, the storage costs for various data in RLWE-based IPFE, including the message vector x , function f , master secret key **msk**, master public key **mpk**, FE-encrypted ciphertext c , functional decryption key dk_f , and the function

Table 3: Running Time and Memory Costs of HFE

Bench	Params	Time (ms)	Memory (MB)
FE.Setup	S	22.2	56.19
	M	494.5	59.20
	L	2967.5	67.73
HE.KGen	S	21.3	57.39
	M	62	57.44
	L	258	57.74
HE.Enc _{pk} (msk)	S	47.8	57.61
	M	1048.4	64.11
	L	6192	84.77
FE.Enc(mpk, x)	S	23.9	62.18
	M	520.9	63.78
	L	3096	77.26
HE.Enc _{pk} (f)	S	48.2	68.87
	M	1050.3	69.56
	L	6208.2	76.32
KDerEval	S	1054.7	68.68
	M	35720.6	72.43
	L	335695.6	112.72
HE.Dec _{sk} (c_{dk})	S	0.4	75.46
	M	1	79.22
	L	3.1	113.12
FE.Dec(dk_f, c_x)	S	5	77.98
	M	68.3	81.35
	L	247.8	112.96

* KDerEval : HE.Eval_{evk}(FE.KDer, (c_{msk}, c_f))

result $f(x)$, are provided for three parameter sets. Note that the sizes of the message and function vectors are equal. As expected, increasing the vector length from 50 to 1000 and enlarging the plaintext and ciphertext modulus leads to higher storage costs. However, the rate of increase is more pronounced for master keys and ciphertexts due to the significant difference between the ciphertext and plaintext modulus. For example, comparing benchmark (**S**) and (**L**), the size of msk grows from 7.73KB to 414.39KB, mpk increases from 24.83KB to 2,665.92KB, and c expands from 24.83KB to 2,665.94KB. Additionally, the ciphertext expansion rate ($\frac{c}{m}$) is approximately $225\times$ and $1353\times$ for benchmark (**S**) and (**L**), respectively. Overall, HFE is $21.8\times$ slower than RLWE-based IPFE for benchmark (**S**), $32.8\times$ slower for benchmark (**M**), and $54.1\times$ slower for benchmark (**L**), while requiring $2.11\times$ more memory for benchmark (**S**), $2.12\times$ for benchmark (**M**), and $2.41\times$ for benchmark (**L**).

Similarly, as presented in Table 5, the storage costs for HFE include the same RLWE-based IPFE data as well as additional BGV components: secret key sk , public key pk , evaluation keys evk , homomorphically encrypted master secret key c_{msk} , homomorphically encrypted message c_x , plaintext function p_f , homomorphically encrypted function c_f , and homomorphically encrypted functional decryption key c_{dk_f} . These costs are provided for three parameter sets. As expected, the storage costs for x , f , msk, mpk, and $f(x)$ in HFE are nearly identical to RLWE-based IPFE, with minor variations due to benchmarking conditions that are negligible.

When examining the additional storage costs introduced by the BGV scheme, for benchmark (**S**), the total storage for sk and pk is less than 2KB, while evk requires 26.44KB. The storage costs for c_{msk} , c_x , c_f , and c_{dk_f} are 47.07KB, 24.26KB, 47.06KB, and 1.32KB, respectively. For benchmark (**L**), the total storage for sk and pk is less than 6.5KB, while evk requires 291.63KB. The storage costs for c_{msk} , c_x , c_f , and c_{dk_f} are 5, 203.88KB, 2, 602.64KB, 5, 203.82KB, and 5.58KB, respectively. Similar to RLWE-based IPFE, the results show that the rate of size increase is higher for master keys and

ciphertexts due to the significant difference between the ciphertext and plaintext modulus. Overall, HFE requires $3.11\times$ more storage than RLWE-based IPFE for benchmark (S), $2.80\times$ for benchmark (M), and $2.84\times$ for benchmark (L), with the ciphertext expansion rate $\left(\frac{c_x}{x}\right)$ reaching $220\times$ for small vectors and $1321\times$ for large vectors.

The comparison between the RLWE-based IPFE scheme and HFE highlights important trade-offs in security, performance, and resource consumption. While HFE introduces additional overheads, these costs are justified by its ability to eliminate the need for a trusted entity, providing a fully trustless and secure solution.

Table 4: Storage Costs of RLWE-based IPFE

Name	S	Size (KB)	
		M	L
m	0.11	0.99	1.97
f	0.11	0.99	1.97
msk	7.73	118.68	414.39
mpk	24.83	529.48	2665.92
c	24.83	529.54	2665.94
dk_f	0.45	0.65	0.98
$f(m)$	0.01	0.01	0.02

Table 5: Storage Costs of HFE

Name	S	Size (KB)	
		M	L
m	0.11	0.99	1.97
f	0.11	0.99	1.97
msk	7.71	118.26	414.04
mpk	24.26	515.67	2602.61
sk	0.44	0.59	0.88
pk	1.32	2.46	5.59
evk	26.44	78.63	291.63
c_{msk}	47.07	1032.79	5203.88
c_m	24.26	515.66	2602.64
c_f	47.06	1032.77	5203.82
c_{dk_f}	1.32	2.45	5.58
dk_f	0.45	0.64	1.05
$f(m)$	0.01	0.01	0.02

9 Conclusions and Future Directions

As this article combines two major modern encryption schemes, namely HE and FE, we provided an in-depth analysis of their definitions, constructions, and related security notions. This extensive background not only gives all the tools to fully comprehend the field but also highlights the significance of this work.

In the introduction, we identified two issues in the classical FE scenario. The first one, the problem of trust in an authority, was the central question and the motivation of our work. To address it, we developed a homomorphic functional encryption (HFE) scheme that takes advantage of the security provided by HE to ensure privacy in an untrusted environment. We showcase the validity of our approach by providing a formal definition of our construction, an analysis of its security, and a protocol supported by practical experiments. The second problem was the question of function privacy, or how to ensure confidentiality of the functional request. The HFE scheme we developed guarantees this feature naturally, without requiring additional computations or assumptions. This underlines the possibilities offered by the combination of HE and FE. Below, we propose future directions to extend the capabilities of our construction.

Future Direction. The protocol we designed eliminates the need for a trusted authority while allowing function-blindness. However, the HFE construction has the potential to support an even wider range of features. In particular, we believe it can serve as a differentially private (DP) FE scheme, ensuring the privacy of individuals in scenarios where a large volume of data is collected from a plethora of users. Originally formalized by Dwork *et al.* [DMNS06], it has been naturally extended to the realm of FE in 2022 by Bakas *et al.* [BMD22] who proposed the first FE scheme achieving DP. Following, Zalonis *et al.* [ZASH24] introduced *noisy* FE – a specific FE paradigm which underpins a

systematic DP construction for most FE schemes. This notion of noisy FE seems to be compatible with our construction, and could benefit from a homomorphic evaluation using an *approximate* HE scheme, such as CKKS [CKKS17].

References

- [AAUC18] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), July 2018.
- [ABDCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 733–751, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
- [ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 552–582, Cham, 2019. Springer International Publishing.
- [ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In *IACR International Workshop on Public Key Cryptography*, pages 128–157. Springer, 2019.
- [ACGU20] Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In *Advances in Cryptology – ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III*, page 467–497, Berlin, Heidelberg, 2020. Springer-Verlag.
- [ADCM12] Michel Abdalla, Angelo De Caro, and Karina Mochetti. Lattice-based hierarchical inner product encryption. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology – LATINCRYPT 2012*, pages 121–138, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 21–40, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 601–626, Cham, 2017. Springer International Publishing.
- [AGT21] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In *Theory of Cryptography Conference*, pages 224–255. Springer, 2021.
- [ANM26] Hossein Abdiniasibfar, Camille Nuoskala, and Antonis Michalas. The hhe land: Exploring the landscape of hybrid homomorphic encryption. *ACM Computing Surveys*, 58(12):1–35, 2026.

- [BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 67–98, Cham, 2017. Springer International Publishing.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.
- [BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
- [BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 470–491, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
- [BM20] Alexandros Bakas and Antonis Michalas. Multi-input functional encryption: Efficient applications from symmetric primitives. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1105–1112, 2020.
- [BM23] Alexandros Bakas and Antonis Michalas. It runs and it hides: A function-hiding construction for private-key multi-input functional encryption. *Cryptology ePrint Archive*, Paper 2023/024, 2023.
- [BMD22] Alexandros Bakas, Antonis Michalas, and Tassos Dimitriou. Private lives matter: A differential private functional encryption scheme. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*, CODASPY '22, page 300–311, New York, NY, USA, 2022. Association for Computing Machinery.
- [Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [BRS13a] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 461–478, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [BRS13b] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, pages 255–275, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [BS15] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 306–324, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography*, pages 253–273, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [CCP⁺24] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the IND CPA-d security of exact FHE schemes. *Cryptology ePrint Archive*, Paper 2024/127, 2024.
- [CDSG⁺18a] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 703–732. Springer, 2018.
- [CDSG⁺18b] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 703–732, Cham, 2018. Springer International Publishing.
- [CDSG⁺20] Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In *Annual International Cryptology Conference*, pages 747–775. Springer, 2020.
- [CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, Jan 2020.
- [CHK⁺18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 360–384, Cham, 2018. Springer International Publishing.
- [CHL20] Sébastien Canard, Adel Hamdi, and Fabien Laguillaumie. Blind functional encryption. In Weizhi Meng, Dieter Gollmann, Christian D. Jensen, and Jianying Zhou, editors, *Information and Communications Security*, pages 183–201, Cham, 2020. Springer International Publishing.
- [CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.
- [CLT18] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p . In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 733–764, Cham, 2018. Springer International Publishing.
- [CRS⁺23] Valerio Cini, Sebastian Ramacher, Daniel Slamanig, Christoph Striecks, and Erkan Tairi. (inner-product) functional encryption with updatable ciphertexts. *Journal of Cryptology*, 37(1):8, Dec 2023.
- [CSBB24] Marina Checri, Renaud Sirdey, Aymen Boudguiga, and Jean-Paul Bultel. On the practical cpad security of “exact” and threshold fhe schemes and libraries. In *Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part III*, page 3–33, Berlin, Heidelberg, 2024. Springer-Verlag.

- [DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [DOT18] Pratish Datta, Tatsuki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 245–277, Cham, 2018. Springer International Publishing.
- [DPY23] Pratish Datta, Tapas Pal, and Shota Yamada. Registered fe beyond predicates:(attribute-based) linear functions and more. *Cryptology ePrint Archive*, 2023.
- [FFM⁺23] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 98–133. Springer, 2023.
- [FT18] Xiong Fan and Qiang Tang. Making public key functional encryption function private, distributively. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 218–244, Cham, 2018. Springer International Publishing.
- [FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, Paper 2012/144, 2012.
- [Gay16] Romain Gay. Functional encryption for quadratic functions, and applications to predicate encryption. *Cryptology ePrint Archive*, Paper 2016/1106, 2016.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC ’09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.
- [GGG⁺14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 578–602, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- [GHR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: removing private-key generator from ibe. In *Theory of Cryptography Conference*, pages 689–718. Springer, 2018.
- [GV23] Robin Geelen and Frederik Vercauteren. Bootstrapping for bgv and bfv revisited. *Journal of Cryptology*, 36(2):12, Mar 2023.
- [HS00] Martin Hirt and Kazuo Sako. Efficient receipt-free voting based on homomorphic encryption. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, pages 539–556, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

- [KLM⁺18] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 544–562, Cham, 2018. Springer International Publishing.
- [KPP22] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. Approximate homomorphic encryption with reduced approximation error. In Steven D. Galbraith, editor, *Topics in Cryptology – CT-RSA 2022*, pages 120–144, Cham, 2022. Springer International Publishing.
- [LM21] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 648–677, Cham, 2021. Springer International Publishing.
- [McE78] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network (DSN) Progress Report*, 44:114–116, 1978.
- [MKMS22a] Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimani. Efficient lattice-based inner-product functional encryption. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 163–193, Cham, 2022. Springer International Publishing.
- [MKMS22b] Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimani. Efficient lattice-based inner-product functional encryption. In *IACR International Conference on Public-Key Cryptography*, pages 163–193. Springer, 2022.
- [MZ17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.
- [NAM24] Camille Nuoskala, Hossein Abdinasibfar, and Antonis Michalas. SPADE: Digging into selective and Partial DEryption using functional encryption. In *Security and Privacy in Communication Networks (SecureComm)*, Cham, 2024. Springer Nature Switzerland.
- [NPP23] Dinh Duy Nguyen, Duong Hieu Phan, and David Pointcheval. Verifiable decentralized multi-client functional encryption for inner product. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 33–65, Singapore, 2023. Springer Nature Singapore.
- [NPS25] Ky Nguyen, David Pointcheval, and Robert Schädlich. Dynamic decentralized functional encryption: generic constructions with strong security. In *IACR International Conference on Public-Key Cryptography*, pages 344–379. Springer, 2025.
- [PD21] Tapas Pal and Ratna Dutta. Attribute-based access control for inner product functional encryption from lwe. In Patrick Longa and Carla Ràfols, editors, *Progress in Cryptology – LATINCRYPT 2021*, pages 127–148, Cham, 2021. Springer International Publishing.
- [RD78] Ronald L. Rivest and Michael L. Dertouzos. On data banks and privacy homomorphisms, 1978.

- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.
- [SPTP⁺21] Sinem Sav, Apostolos Pyrgelis, Juan Ramon Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. Poseidon: Privacy-preserving federated neural network learning. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. INTERNET SOC, 2021.
- [SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *Theory of Cryptography*, pages 457–473, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 457–473. Springer, 2005.
- [Tom19] Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 459–488, Cham, 2019. Springer International Publishing.
- [Tom23] Junichi Tomida. Unbounded quadratic functional encryption and more from pairings. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 543–572, Cham, 2023. Springer Nature Switzerland.
- [TÜ24] Erkan Tairi and Akin Ünal. Lower bounds for lattice-based compact functional encryption. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 249–279, Cham, 2024. Springer Nature Switzerland.
- [TZR⁺22] Han Tian, Chaoliang Zeng, Zhenghang Ren, Di Chai, Junxue Zhang, Kai Chen, and Qiang Yang. Sphinx: Enabling privacy-preserving online learning over the cloud. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2487–2501. IEEE, 2022.
- [WWSG23] Ming Wan, Geng Wang, Shi-Feng Sun, and Dawu Gu. Function-hiding zero predicate inner product functional encryption from pairings. In Elias Athanasopoulos and Bart Mennink, editors, *Information Security*, pages 516–534, Cham, 2023. Springer Nature Switzerland.
- [ZASH24] Jasmin Zalonis, Frederik Armknecht, and Linda Scheu-Hachtel. Differentially private functional encryption. *Proceedings on Privacy Enhancing Technologies*, 2024:509–530, 04 2024.