Blind Functional Encryption

Sébastien Canard, Adel Hamdi, Fabien Laguillaumie

Blind Functional Encryption

Sébastien Canard[1], Adel Hamdi[1,2], and Fabien Laguillaumie[2]

[1] Orange Labs, Applied Crypto Group, Caen, France.
[2] Université Claude Bernard Lyon 1, LIP, Lyon, France.

Abstract. Functional encryption (FE) gives the power to retain control of sensitive information and is particularly suitable in several practical real-world use cases. Using this primitive, anyone having a specific functional decryption key (derived from some master secret key) could only obtain the evaluation of an authorized function f over a message m, given its encryption. For many scenarios, the data owner is always different from the functionality owner, such that a classical implementation of functional encryption naturally implies an interactive key generation protocol between an entity owning the function f and another one managing the master secret key. We focus on this particular phase and consider the case where the function needs to be secret.
In this paper, we introduce the new notion of *blind functional encryption* in which, during an interactive key generation protocol, the master secret key owner does not learn anything about the function f. Our new notion can be seen as a generalisation of the existing concepts of blind IBE/ABE. After a deep study of this new property and its relation with other security notions, we show how to obtain a generic blind FE from any *non-blind* FE, using homomorphic encryption and zero-knowledge proofs of knowledge. We finally illustrate such construction by giving an efficient instantiation in the case of the inner product functionality.

1 Introduction

With the growth of online activities, multiple data (confidential emails, employment contracts, bank transactions, etc.) are transmitted and stored over different external platforms. A ruthless competition between several actors is ongoing in order to offer particular services, based on those data, thus answering positively to an increasing demand. For example, one could subscribe to a *malware detection service* (or a *spam filter*) that aims to identify bad patterns over some incoming messages and, at best, to reject them. In a different use case, a company or an institution specialized in machine learning algorithms could find interest to obtain some specific data from a data owner to improve its algorithms: individuals with specific characteristics related to e.g., healthcare, or companies with some specific kind of data for e.g., threats detection related to Intranet/Internet browsing. At the same time, several concerns about the security and privacy of manipulated data bring new challenges to those organizations in this context. Encryption mechanism is one enabler to achieve the compliance and data security/privacy that is required in today's security interest. However, conciliate

data confidentiality and functionality could be a hard task by using basic *all-or-nothing* approach of traditional encryption schemes, where no computation are possible, except by decrypting the data itself, then decreasing the obtained security. From a higher perspective, we consider a scenario with an entity that try to get in clear a function over some encrypted data.

In recent years, *Fully Homomorphic Encryption* (FHE) [20] and *Functional Encryption* (FE) [8] arise as a general and very promising framework that gives the flexibility and the possibility to retain control of leaked information. Where FHE permits to delegate some computation over sensitive data to third parties, FE gives the power from an encryption of a message m and functional decryption key sk_f for a certain function f, to obtain in clear the evaluation $f(m)$ and no additional information.

Motivation. In a FE scheme, the function decryption key sk_f is derived from a master secret key msk and the function f. The master key owner is then very powerful and (even if mainly separate) is most of the time close to the data owner. It follows that in most use cases, the functional key generation protocol is interactive between the owner managing the master secret key msk and the owner of the algorithm knowing the function. While the natural approach to obtain sk_f is to send f to the master secret key's owner, we give amongst other concerns interest to a situation when the evaluation function f could be sensitive. In the malware detection example, it corresponds to the market compliance defined in e.g. [13] which shows the sensitivity of the rules given by the security editor. In the data analytics scenario, the underlying machine learning algorithm to better detect a specific disease or a malware is sometimes linked to some very specific and rare know-how. Hence, it could be relevant and crucial to *blind* the underlying structure to the master secret key owner.

Our Contributions. More precisely, we provide in this paper the following three main contributions.
Contribution 1: general definition of blind interactive FE. For real-life applications, the functional key extraction *is* interactive. The authority $\mathcal{AUT}$ that controls the msk must get the function f in some way. This lead us to consider in this work an *interactive* functional key generation phase. The definition of IFE is then adapted and similar to the one of FE (i.e we maintain Setup, Enc, Dec and the correctness condition), except that we replace the KeyGen algorithm by an IKeyGen two-party protocol between $\mathcal{AUT}$ and $\mathcal{U}$. The result of the interaction is a functional key sk_f for $\mathcal{U}$ and some output defined by the view of $\mathcal{AUT}$. In addition, we provide some adapted security definition from FE to the IFE case. In particular, the message-privacy (MP) property asks that no additional information about m is produced by the system except of $f(m)$, while the function-privacy (FP) asks that the functional key does not leak additional information about f. We show how to adapt these existing security definitions from FE to the IFE case. Then, our new notion of *blindness* is inspired by the notions of blind signatures [26] or blind identity-based encryption (IBE) [24, 11].

Intuitively, it means that a curious $\mathcal{AUT}^\star$ cannot link a functional key to an interaction it had with an honest user $\mathcal{U}$. Even if it looks related to the FP security, we show in the sequel that our notion of blindness is different (and complementary) from FP in the general case.

Contribution 2: generic construction of blind IFE from any FE. A possible approach to derive generically an interactive FE would be to use a secure two-party computation of the IKeyGen protocol. We insist that such an approach *does not* achieve the blindness property we are interested in. Indeed, although the authority does not learn the user's input with 2PC, it could make the functional keys output by two users in the blindness security game depend on the function in different ways. This is possible by using for example two different master keys. Here is an overview of the construction (see Sec. 3): Our approach starts from an existing FE scheme for a class of function F and upgrades it to a blind IFE scheme from the same class F, by only modifying the KeyGen algorithm. $\mathcal{U}$ starts by *encrypting* an encoded version of some function f with a Fully Homomorphic scheme FHE under her own key and sends the ciphertext C_f to $\mathcal{AUT}$. With msk, the party $\mathcal{AUT}$ homomorphically evaluates the circuit $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ using the FHE.Eval algorithm on C_f, then sends back a ciphertext C_{sk_f} of the corresponding functional key sk_f. $\mathcal{U}$ can now decrypt with her (FHE) secret key the received ciphertext and recover sk_f. Thereby, the FHE *blinds* to $\mathcal{AUT}$ both the function f and the key sk_f. However, this basic protocol is insecure since each entity could cheat on its input, hence we provide some modifications using Zero-Knowledge Proofs of Knowledge (ZKPoK) mechanisms to ensure correct behaviour and to prevent from getting some unauthorized functions. With this considerations, we are able to obtain a feasibility result on the construction of blind IFE scheme.

Contribution 3: specific construction for IPFE. Many applications, such as data mining or statistical computation need as subroutines inner product evaluation. That is why several [2, 5, 17] IPFE constructions have recently been proposed. Most of known schemes extract functional keys of the same shape: $(y, \langle s, y\rangle)$ where $s, y \in \mathbb{Z}_p^\ell$ for a (large) prime p, where $\langle x, y\rangle := \sum_{i=1}^{\ell} x_i \cdot y_i$ is the inner product of $x \in \mathcal{R}^\ell$ and $y \in \mathcal{R}^\ell$ for some ring $\mathcal{R}$. Our contribution is to give an efficient two party protocol computing these functional keys with the blindness property, and which can be used in the constructions whose functional key is an inner product. Hereafter, we modify the construction from [22] by using the Castagnos-Laguillaumie (CL) linear homomorphic encryption from [16] since we need inner product computed in $\mathbb{Z}_p$. We can then directly embed this protocol into secure DDH-based schemes like those of [2, 5] or in the CL-based protocol from [17]. We develop in Sec. 4 an IKeyGen protocol that implements the KeyGen algorithm in order to build a blind IPFE scheme.

Related Work and discussion. The notion of interactive key generation is considered in the case of *Accountable-Authority* Identity-Based Encryption (IBE) in [23]. The first consideration of *blindness* for IBE appears in the work of Green and Hohenberger in [24] followed by of Camenisch et al. [11] where it was used as a building block for respectively a *simulatable oblivious transfer* and a *public key encryption with oblivious keyword search*. In [28], an adaptation is proposed

for the case of the Attribute-Based Encryption (ABE) primitive. More recently, [19] consider a variant of blind IBE to resolve the *key escrow* problem. As far as we know, no such study has been done for the more general case of functional encryption. *Controlled* functional encryption [27] is also a variant of FE with an interactive behaviour. While similar to our general approach of hiding the function to the authority, the model is different from ours. [3]

Private function evaluation (PFE) [1] is closely related to our problem. In PFE, a party P_1 holds an input x while another party P_2 holds a circuit C_g describing a function g; the goal is for one (or both) to learn the result $g(x)$. Our blind IFE could be seen as a PFE with the additional property of blindness, which is not automatically guaranteed by a generic PFE. Eventually, the security requirements could be defined in terms of simulatability which informally enable to design an ideal functionality that captures previous properties (message-privacy, function-privacy or blindness) and consider interdependent executions with other protocols while preserving the main security characteristics. However, we took the classical approach to provide a natural generalization of the blindness property, as well as the classical security notions for FE (message/function privacy) in the presence of an interactive key generation protocol. This has the benefits to only adapt existing definition by adding some interactive oracles, and avoid eventually some subtle negative results, as in the context of simulation-based blind signature [3]. In addition, our solution encompasses the existing definitions for IBE/ABE cases [24, 11, 25, 28] presented in the literature.

2 Blind Interactive Functional Encryption

Based on the known definitions of FE [8], we formally define our new notion of *blind interactive* functional encryption. Our goal is twofold. We first want to capture the situation of a user holding a function f and asking an authority for a corresponding functional key sk_f during an interactive protocol. We then consider the case where the user wants to protect the function f from the authority. In the sequel, we introduce the notion of *interactive* FE with the new security notion of *blindness*. In addition, we discuss some related security properties.

2.1 Syntactic Definitions for Interactive FE

We set for the rest of the paper two specific parties: an *authority* denoted by $\mathcal{AUT}$ and a *user* denoted by $\mathcal{U}$. For a $\lambda \in \mathbb{N}$, fix an arbitrary set of functions F represented by a poly-sized family of circuits $\{F_\lambda\}_{\lambda\in\mathbb{N}}$ and a *message* space M, where each $m \in M \subseteq \{0,1\}^*$ is represented by a string input of any $f \in F$. A *public key interactive functional encryption* is defined as follows.

[3] There are two parties in our model where the master secret key owner is the only party to provide functional keys. In [27], it is only possible to produce functional keys that depends on the ciphertext and is only used once, while we consider multiple users, functional keys and ciphertexts.

Definition 1 (Public key IFE). *Let $\lambda \in \mathbb{N}$. An interactive functional encryption scheme for F consists of a tuple* $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *where,*

- $\mathsf{Setup}(1^\lambda)$ *is a PPT algorithm that takes as input a security parameter* 1^λ *and outputs a master secret key* msk *and a master public key* mpk.
- $\mathsf{IKeyGen}(\mathcal{AUT}(\mathsf{msk}), \mathcal{U}(\mathsf{mpk}, f))$ *is a 2-party interactive protocol between an authority* $\mathcal{AUT}$ *with input the master secret key* msk *and a user* $\mathcal{U}$ *with inputs a master public key* mpk *and a function* $f \in F$. *The output of this protocol is, on the authority's side* $\mathtt{Output}(\mathcal{AUT})$ *and on the user's side,* sk_f.
- $\mathsf{Enc}(\mathsf{mpk}, m)$ *is a PPT algorithm which takes as input the master public key* mpk *and a message* $m \in M$, *and returns a ciphertext* c.
- $\mathsf{Dec}(\mathsf{mpk}, sk_f, c)$ *is a PPT algorithm which takes as input a master public key* mpk, *a functional key* sk_f *and a ciphertext* c *and outputs a string* z.

For correctness, for all $f \in F$ and $m \in M$, given $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, sk_f *resulting from* $\mathsf{IKeyGen}$ *protocol between (honest)* $\mathcal{AUT}$ *and* $\mathcal{U}$ *and* $c \leftarrow \mathsf{Enc}(\mathsf{mpk}, m)$, *we require* $\mathsf{Pr}\left[\mathsf{Dec}(\mathsf{mpk}, sk_f, c) = f(m)\right] \geq 1 - \mathsf{negl}(\lambda)$.

The above definition can easily be adapted to the *private-key* setting. Notice that functional encryption denoted by $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ falls as a particular case, i.e there is a non-interactive KeyGen algorithm executed by the owner of msk ($\mathcal{AUT}$ in our context) which outputs a functional key sk_f. In addition, our generic conversion will start from a FE scheme with a determined KeyGen algorithm. Since it takes (the description of) f and msk as inputs, it will be necessary to specify the *size* of the circuit that computes the function f in addition to the *size* and *depth* of the circuit computing KeyGen.

A Trivial Example. In the following, we present a simple IFE that one can obtain from any FE. Given $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ following Definition 1, it is easy to define a *trivial* interactive IFE scheme as $\mathsf{Trivial.IFE} := (\mathsf{Setup}, \mathsf{Trivial.IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{Trivial.IKeyGen}$ protocol is defined in Fig. 1. Here, the user simply asks the msk owner's (i.e $\mathcal{AUT}$) to generate the functional key sk_f. In particular, this protocol corresponds to the most common implementation for real-life use-cases of FE as discussed in the introduction.

While it is simple, it is interesting to notice that in $\mathsf{Trivial.IFE}$, the user does not learn any information about msk. In particular, the intuition is to conjecture that the resulted IFE scheme will inherit the message-privacy of the FE scheme. We prove this fact in Prop. 2, but this observation also gives the intuition of the notion of the *leak-freeness* property that we will define in Sec. 2.3. On the other hand, blindness is *not* guaranteed by construction since $\mathcal{AUT}$ *learns* f.

Validity of sk_f. One issue of this trivial example is that the authority may have sent to the user a fake key sk_f. Thus, the latter should have a way to verify its validity. One solution was given for interactive blind IBE [24, 11]. They propose to encrypt a polynomial number of random messages with id, then try to decrypt using the obtained identity-related key. A first idea can be to proceed similarly, which works quite well in the public key setting and in the case of (indexed) functions of the form of $f_k(m, y) := m \iff R(k, y) = 1$ where R is a publicly

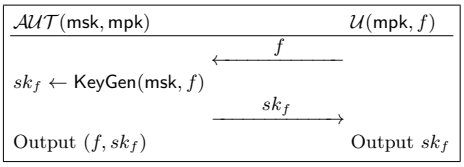

Fig. 1. Trivial.IKeyGen

known relation. However, in the general case, this method may obviously not convince a user of the validity of the sk_f, and is definitely not possible in the private key setting. We then propose to make use of a Zero-Knowledge Proof of Knowledge ZKPoK, generated by the authority to prove that it has correctly computed sk_f, as $\pi \leftarrow \mathsf{ZKPoK}\{\mathsf{msk} : sk_f = \mathsf{KeyGen}(\mathsf{msk}, f)\}$. We stress that considering the validity of sk_f is an *additional* requirement and we can have this property using another approach. However, when dealing with blindness, having ZKPoK could help in order to *force* $\mathcal{AUT}$ to assure that sk_f is well formed.

2.2 High-Level View of Security Properties

An interactive FE must first verify a message-privacy property. This will be discussed in Section 2.3. Then, we consider our new notion of blindness and discuss other properties. We first analyse what the authority could learn.

Output of the authority. The fact that we want to hide the function to the authority is at first related to the authority's view of the interactive protocol. Indeed, intuitively, the *best* case (hiding f *and* sk_f) would be an authority which does not learn anything more that what it already knew before the interaction. Recall that the authority, by definition, can deduce $\mathtt{Output}(\mathcal{AUT})$ from its own view $\mathtt{View}_{\mathcal{AUT}}(\mathsf{msk}, f) := (\mathsf{msk}, r; m^1, \ldots, m^t)$, where r some random elements, and m^j the jth message that it received from some interaction.

1. **Considering f in the output.** To ensure a notion of *blindness* of the key generation algorithm, the authority cannot obtain from the received messages m_j, or more generally, from $\mathtt{View}_{\mathcal{AUT}}(\mathsf{msk}, f)$, any information about the user's choice of the function. A standard solution, as in the context of blind signature [26], is to ask the authority *to link* a functional key sk_f generated during an interaction to the corresponding function f. Informally, the adversary runs two random sequential executions of the protocol with two users and is asked to link the produced functional keys to each user. We will call this notion *blindness* which is, to the best of our knowledge new in the general context of functional encryption. We treat this security notion in Sec. 2.4. In particular, having f, or some information about f during one execution (i.e is one of the m_j), as for the construction of Trivial.IFE (see Ex. 2.1), gives a way to find the user's choice, thereby breaking blindness.
2. **Considering sk_f in the output.** The functional key sk_f is used to decrypt ciphertexts c_m of some messages m in order to obtain values $f(m)$. For the authority, which is in possession of msk, it is possible to encrypt any message

m of its choice. If sk_f can be deduced from $\texttt{View}_{\mathcal{AUT}}$, then the authority can learn arbitrary information about f (every $f(m)$ of its choice). This last observation remains true even if we start from a *function-private* FE where sk_f have some *hiding* property and does not leak any information about the function f. We deduce that having sk_f in the authority's view breaks the blindness requirement of the last paragraph since it easy to distinguish two sequential interactions. Because of this access to an unlimited evaluation of the function f, we remarked that the same problem arises in the context of function-private (FP) *public key* functional encryption [10, 7] where hiding information about f in sk_f gives the same restrictions. We give a formal treatment of function-privacy for IFE in the full version.

Finally, we provide in Sec 2.5 a discussion about possible relations between FP and blindness where we prove that there are in fact two separate notions.

2.3 Message-Privacy for Interactive FE

Adaptation from FE. The basic security consideration for functional encryption is related to the standard notion of semantic security in presence of different functional keys [8]. As it is usually done, we consider the adaptive form of message-privacy with multiple messages and multiple functional keys. Our notion of message privacy is a direct adaptation of this classical notion when we have to consider interactive oracles. We refer to Def. 8 in the appendix for the formal definition. Next, consider the IFE with the Trivial.IKeyGen from Example 2.1. The user sends f and the authority generates sk_f using msk. The following proposition is immediate and the proof is given in the full version.

Proposition 2. *The* Trivial.IFE *of Example 2.1 is message-private if the underlying* FE *is message-private.*

There are two different ways to prove that an interactive FE is message-private. Obviously, the direct way which shows that a protocol fulfils the Definition 8. Another possibility relies on the notion of *leak-freeness* that we will present next.
Leak-Freeness. Recall that in Trivial.IFE of Ex. 2.1, the curious user does not learn any information about the master secret key msk that could help her to break the MP security game of the FE scheme. Regarding blindness, we note however that the user has to hide to the authority its inputs, but could cause the protocol to leak additional information about msk. In particular, when building a message-private IFE, one could hope to only get the information that could be obtained from a natural Trivial.IFE. Informally, we have to compare the different information that could be obtained from the proposed interactive key generation and the trivial implementation of FE, i.e the Trivial.IFE.

Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a message-private scheme. Inspired by the work done for IBE [24, 11], we generalize the notion of *leak-freeness* for functional encryption. Such notion aims at providing a condition to *preserve* from learning any additional informations, due to the interactive key generation, that could break the message-privacy. Informally, it makes possible to prove that

an IFE.IKeyGen protocol executed with an honest authority does not leak more information than the Trivial.IKeyGen from Example 2.1, with the same honest authority. The main intuition is that such notion can then be used to prove that the resulting interactive functional encryption $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is also message-private. The formal definition of leak-freeness is given in appendix C. The motivation of this notion is given by the following proposition.

Proposition 3. *Let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *be a message-private secure* FE *scheme. Let* $\mathsf{IFE} := (\mathsf{Setup}, \mathsf{Enc}, \mathsf{IKeyGen}, \mathsf{Dec})$. *If* IFE.IKeyGen *is leak-free with respect to* KeyGen*, then* IFE *is message-private.*

Due the lack of space, the proof of this proposition is not provided here but it could be seen as a generalization of the result in the IBE case presented in [24]. This proposition is used to prove the security of our generic construction.

2.4 Blindness for Interactive FE

In this section we formally define our new *blindness* property. Intuitively, following the usual definition for blind signatures [26], blindness means that the authority cannot link a functional key to an interaction it had with an honest user. This is clearly related to the information that the authority has at the end of the key generation protocol, namely $\mathtt{Output}(\mathcal{AUT})$.

It is possible to define a unique notion of blindness independently for both the private and public key settings. This situation is simulated by an adversary who can choose maliciously the parameters but follows the protocol. His aim is to decide which of two chosen functions f_0, f_1 has been used to generate the functional keys sk_{f_0} and sk_{f_1} in two sequential executions with an honest user $\mathcal{U}$. We call this notion *blindness* and corresponds to a variant of the selective-failure blindness security considered in [24, 11] for IBE, which adds the following property: the authority cannot cause the protocol to fail in a manner dependent on the user's choice. This additional security requirement was used in order to build oblivious transfer [24] or searchable encryption [11]. Here we consider basic definitions and leave extensions for further applications.

We introduce the interactive oracle $\mathsf{IKeyGen}(\cdot, \mathcal{O}(\mathsf{mpk}, f))$ in which the adversary plays the role of the authority and only obtains his own output. In the game below, we write $\mathcal{A}^{\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(f_0))/\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(f_1))}$, which mean that $\mathcal{A}$ can query each oracle only once (hence the notation $\mathsf{IKeyGen}^{(1)}$) and that the two oracles can be invoked in an arbitrary order.

Definition 4 (Blindness). *Let* $b \in \{0, 1\}$. *An* IFE *is* ***blind****, if every adversary* $\mathcal{A}$ *has a negligible advantage* $|\mathsf{Pr}[b' = b] - 1/2|$ *in the following experiment*

1. $(\mathsf{mpk}, f_0, f_1, st_{find}) \leftarrow \mathcal{A}^{\mathsf{Setup}(\cdot)}(find, 1^\lambda)$.
2. $st_{issue} \leftarrow \mathcal{A}^{\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(\mathsf{mpk},f_b))/\mathsf{IKeyGen}^{(1)}(\cdot,\mathcal{O}(\mathsf{mpk},f_{1-b}))}(issue, st_{find})$*, the step produces at the end of the executions local outputs (possibly undefined* $\perp$*)* sk_{f_b} *and* $sk_{f_{1-b}}$ *respectively.*
3. *If* $sk_{f_0} = \perp$ *or* $sk_{f_1} = \perp$*, set* $(sk_{f_0}, sk_{f_1}) = (\perp, \perp)$.

4. $b' \leftarrow \mathcal{A}(guess, sk_{f_0}, sk_{f_1}, st_{issue})$.

This definition can easily be adapted to the private key setting.
Remark. It is important to note, as in the context of blind signatures, that any information about sk_f that can be deduced during the interaction from the $\texttt{Output}(\mathcal{AUT})$ leads our definition to fail. Indeed, for example if $\mathcal{A}$ gets sk_f in the end of the interaction, it will obviously win the game by just interacting with one of the two oracles. In fact, any *left-or-right* definition would fail, since during the interaction there is always a way to distinguish between two keys/interactions. This difficulty comes for the inherent capabilities of the FE scheme. From the encryption of a certain message m such that $f_0(m) \neq f_1(m)$ and an interaction giving sk_{f_b} at the end of one of the two interactions, it is always possible to decrypt and get $f_b(m)$. Since f_0 and f_1 are chosen by $\mathcal{A}$, it seems clear that the blindness implies in particular that the malicious authority does not get information about sk_f and f during (or in the end of) the interaction.

2.5 On the Relationship between Function-Privacy and Blindness

Function-privacy for (I)FE. Several other security properties have been considered for FE in the literature and we will not review all of them. However, we could generalize the known [7, 10, 4] notion of function-privacy which informally states that a functional key sk_f does not give any additional information about the underlying function f, except from what is given by the evaluations over some data being encrypted. We give in the full version a generalization of it in the context of IFE where informally, interactive oracles are added in order to consider potential leakage during the interaction. Since our aim is to present a blind IFE scheme, we only briefly highlight the differences between theses notions.

Depending on the public or private key setting and the presence or not of sk_f in the authority's output, we obtain several (dis)connections between function-privacy and blindness security properties. Informally, this is due to the *nature* of the considered options. Indeed, the FP security asks *any adversary* which does not have necessarily an access to an encryption oracle, to obtain unwanted information about the function f from sk_f and eventually the interaction. The blindness security game concerns *an authority* with the capability of encrypting arbitrary messages using the master key msk. We now give our main theorem which, in a nutshell, says that these two properties are distinct, and then complementary. In the full version of this paper, we provide a detailed proof of this theorem by providing, for each of the resulted six cases a separating construction.

Theorem 5. *Blindness and Function-Privacy properties are mutually separated, for both private-key and public-key* IFE.

3 Generic Construction of Blind IFE from fully homomorphic encryption

We provide in this section a generic construction of a message-private blind interactive functional encryption from any FE, where $\mathcal{AUT}$ does not obtain any

- $\mathsf{IFE.Setup}(1^\lambda)$: Output $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
- $\mathsf{IFE.IKeyGen}(\mathcal{AUT}(\mathsf{msk}), \mathcal{U}(\mathsf{mpk}, f))$ is described in Fig. 3
- $\mathsf{IFE.Enc} = \mathsf{FE.Enc}$
- $\mathsf{IFE.Dec} = \mathsf{FE.Dec}$

Fig. 2. Our generic blind IFE

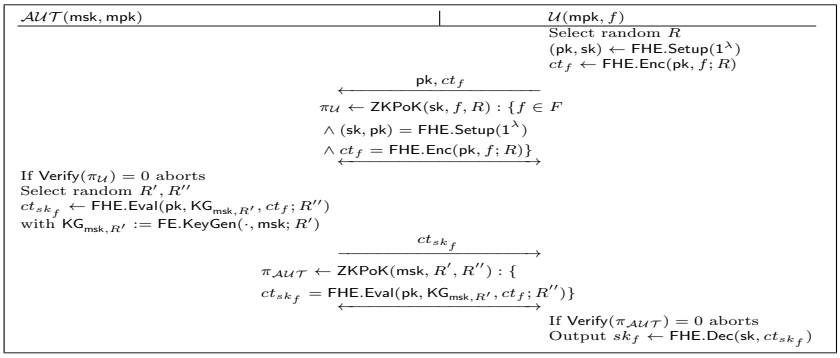

Fig. 3. Our interactive key generation $\mathsf{IFE.IKeyGen}$

information at the end of the interactive key generation. In addition, our concern is to not modify the $\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}$ algorithms but only the KeyGen algorithm.

3.1 Our Generic Construction

Let $\lambda > 0$ be a security parameter and consider a family of functions $F = \{F_{n,\lambda}\}_{n=n(\lambda)}$ whose input size $n(\lambda)$ is polynomial in λ. Suppose that all functions $f \in F$ can be encoded as a $p(\lambda)$-bit string (for a polynomial p). Consider a functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ for this family F. We suppose that $\mathsf{FE.KeyGen}$ is a randomized algorithm that is described by a circuit of logarithmic depth $d(\lambda)$. Consider $\mathsf{FHE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ to be a *CPA-secure* Fully Homomorphic Encryption scheme, where the input encryption algorithm is a bit string with size at least $p(\lambda)$ and supports evaluation of circuits of depth at least $d(\lambda)$. Our interactive blind functional encryption for the class of function F is described in Fig. 2 and 3. Note that encryption and decryption are exactly those of the original FE.

Correctness. By correctness of the FHE scheme, the user $\mathcal{U}$ obtains after decryption $sk_f = \mathsf{KG}_{\mathsf{msk},R'}(f) := \mathsf{FE.KeyGen}(\mathsf{msk}, f; R')$ for certain random R'.

3.2 Security of our Construction

Roughly speaking, the CPA-secure FHE will insure blindness since two interaction transcripts are indistinguishable and the ZKPoK will guarantee that no additional information is leaked from the interaction. Additionally, we will need a notion of *weak-function indistinguishablity* which informally says that any adversary (even if it knows the secret key of the FHE) cannot produce any FHE

ciphertext ct_y, for input y, two functions h_0, h_1 with $h_0(y) = h_1(y)$ such that it could distinguish between $\mathsf{FHE.Eval}(\mathsf{pk}, h_0, ct_y)$ and $\mathsf{FHE.Eval}(\mathsf{pk}, h_1, ct_y)$ (we refer to [6] and the full version of this paper for a precise definition).

Theorem 6. *Consider a message private* FE *in addition to a weak-function indistinguishable CPA-secure* FHE *scheme. If the proofs $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{AUT}}$ are* ZKPoK*, then the* IFE *described in Fig. 2 is message-private and blind.*

Due to space limitation, we only sketch the main ideas of the MP security proof and provide a complete proof of blindness.

Sketched proof of MP. Similarly to the proof [24] for the IBE case, we first prove that the IFE.KeyGen protocol is *leak-free* (see Sec. 3 and Def. C) with respect to FE.KeyGen and by Prop. 3, it implies the IFE is *message-private*. More precisely, we describe a simulator (which does not have msk) that makes use of the extractability of zero-knowledge proofs of knowledge to obtain the function f. Then, having access to some oracle Trivial.IKeyGen providing sk_f, it could simulate a valid interactive IFE.IKeyGen protocol with any adversary by using (i) the homomorphic property of the FHE to generate the ciphertexts and (by evaluating the constant circuit equal to sk_f) (ii) its rewinding capability together with the zero-knowledge property of ZKPoK to simulate the proofs. Remark that in the simulation of the ciphertexts, the adversary cannot notice the difference thanks to the weak-function indistinguishability notion. Finally, we deduce that the IFE.IKeyGen is leak-free w.r.t FE.KeyGen and the result follows.

Proof of Blindness. In the proof of blindness, we have to show that the (sequentially generated) messages exchanged between a malicious authority $\mathcal{AUT}$ and two honest users are completely independent from the functions. We prove it via a sequence of hybrid games. By using the extractability of ZKPoK, we first obtain msk from the interaction with the adversary. This allows us to replace the generation of the functional secret key by the non-interactive version, in a non-detectable way. We finally reduce our problem to the one of the CPA-security of the FHE scheme. In the final game, the malicious authority obtains messages that are independent from the functions of its choice, which imply blindness.

Suppose we have an adversary $\mathcal{A}$ attacking the blindness game. Recall that it chooses the public parameters $(\mathsf{mpk}, \mathsf{msk})$ and two functions f_0 and f_1, then runs two sequential interactions with honest user $\mathcal{U}(\mathsf{mpk}, f_b)$ and $\mathcal{U}(\mathsf{mpk}, f_{1-b})$ respectively where b is a random bit. At the end of the interactions, $\mathcal{A}$ receives the two functional keys (sk_{f_0}, sk_{f_1}) (or $(\perp, \perp)$) corresponding to (f_0, f_1). The goal for $\mathcal{A}$ is to find the bit b with non-negligible probability. We note $\bar{b} := 1 - b$. We prove the blindness property via a sequence of games.

Game 0. This is the original game as in Def. 4. We give more details about each interaction in Fig. 4. We will describe the interaction of the adversary with each oracle user $\mathcal{U}_b$ and $\mathcal{U}_{\bar{b}}$. Lines 1,7,11-12 describe the behaviour of $\mathcal{A}$ during the blindness game and the remaining lines the user's behaviour. Game 1. We modify Game 0 in the following way. In this game, because there are two possible ZKPoK (see line 8) , we know that there exists an extractor $\mathsf{Ext_b}$ (resp. $\mathsf{Ext_{\bar{b}}}$) that can extract the witnesses from π'_b (resp. $\pi'_{\bar{b}}$) and obtain $w^*_b = (\mathsf{msk}^*_b, R'_b, R''_b)$ for each bit $b \in \{0, 1\}$. We add the following quantities for each user in line

Game 0	
1. $(\mathsf{mpk}, f_0, f_1) \leftarrow \mathcal{A}^{\mathsf{Setup}(\cdot)}(1\lambda)$	
2. $(\mathsf{pk}_b, \mathsf{sk}_b) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$	$(\mathsf{pk}_{\bar{b}}, \mathsf{sk}_{\bar{b}}) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$
3. $ct_{f_b} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}_b, f_b; R_b)$	$ct_{f_{\bar{b}}} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}_{\bar{b}}, f_{\bar{b}}; R_{\bar{b}})$
4. $w_b \leftarrow (\mathsf{sk}_b, f_b, R_b)$	$w_{\bar{b}} \leftarrow (\mathsf{sk}_{\bar{b}}, f_{\bar{b}}, R_{\bar{b}})$
5. $m_b := (\mathsf{pk}_b, ct_{f_b})$	$m_{\bar{b}} := (\mathsf{pk}_{\bar{b}}, ct_{f_{\bar{b}}})$
6. $\pi_b \leftarrow \mathsf{ZKPoK}(\mathcal{O}(w_b), \mathcal{A}(m_b))$	$\pi_{\bar{b}} \leftarrow \mathsf{ZKPoK}(\mathcal{O}(w_{\bar{b}}), \mathcal{A}(m_{\bar{b}}))$
7. $(ct'_b, ct'_{\bar{b}}) \leftarrow \mathcal{A}((\pi_b, m_b), (\pi_{\bar{b}}, m_{\bar{b}}))$	
8. $\pi'_b \leftarrow \mathsf{ZKPoK}(\mathcal{A}(w'_b), \mathcal{O}(ct'_b))$	$\pi'_{\bar{b}} \leftarrow \mathsf{ZKPoK}(\mathcal{A}(w'_{\bar{b}}), \mathcal{O}(ct'_{\bar{b}}))$
9. If $\mathsf{Verify}(\pi_b) = 1$	If $\mathsf{Verify}(\pi_{\bar{b}}) = 1$
10. $sk_{f_b} \leftarrow \mathsf{FHE.Dec}(ct'_b, \mathsf{sk}_b)$ else $sk_{f_b} \leftarrow \perp$	$sk_{f_{\bar{b}}} \leftarrow \mathsf{FHE.Dec}(ct'_{\bar{b}}, \mathsf{sk}_{\bar{b}})$ else $sk_{f_{\bar{b}}} \leftarrow \perp$
11. $b' \leftarrow \mathcal{A}(\mathsf{sk}_{f_0}, \mathsf{sk}_{f_1})$	
12. returns 1 iff $b' = b$	

Fig. 4. Blindness experiment.

8: $w_b^* := (\mathsf{msk}_b^*, R_b^{'}, R_b^{''})$ and $w_{\bar{b}}^* := (\mathsf{msk}_{\bar{b}}^*, R_{\bar{b}}^{'}, R_{\bar{b}}^{''})$. The matching condition prevents the adversary to use two different master secret keys. Thanks to the extractability condition, the rewinding techniques of the ZKPoK, it is possible to efficiently extract the corresponding witness, and for the adversary, the success probability remains the same (except with negl. probability). Assuming that the π' are proofs of knowledge, Game 0 is then indistinguishable from Game 1.

Game 2. We modify the Game 1 in the following way. If the master secret keys do not match, i.e $\mathsf{msk}_b^* = \mathsf{msk}_{\bar{b}}^*$, the user oracles in the two interactions abort and we set $(sk_{f_0}, sk_{f_1}) = (\perp, \perp)$. Otherwise, we set $\mathsf{msk} := \mathsf{msk}_b^*$ and instead of decrypting ct'_b (or $ct'_{\bar{b}}$), as in line 10, we exploit the extracted value $\mathsf{msk}, R_b^{'}$ and the $\mathsf{FE.KeyGen}(\mathsf{msk}, \cdot; \cdot)$ algorithm on input $(f_b, R_b^{'})$ (resp. $(f_{\bar{b}}, R_{\bar{b}}^{'})$) to obtain valid functional key(s). We replace line 10. by the new line (depending on b) $sk_{f_b} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, f_b; R'_b)$ and $sk_{f_{\bar{b}}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, f_{\bar{b}}; R'_{\bar{b}})$. If the proof does not fail the oracles return (locally) sk_{f_b} (and $sk_{f_{\bar{b}}}$). Otherwise, it returns $(sk_{f_0}, sk_{f_1}) = (\perp, \perp)$. Finally, we give as in line 11. (sk_{f_0}, sk_{f_1}) to $\mathcal{A}$. Thanks to the correctness of FHE and FE, Game 1 is indistinguishable from Game 2.

Game 3. We change the behaviour of user $\mathcal{U}_1$ and encrypt a randomly chosen function $g_1 \in F$ with size description equal to f_1 with a modified proof of π_1 in the first message and π'_1 in the second one. In more details, there exists a zero-knowledge simulator Sim_1 for π_1 that can simulate the proof of knowledge without knowing the underlying witness. We replace the term in line 3. for $\mathcal{U}_1$ with $ct_{g_1,1} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}_1, g_1; R_1)$, where R_1 is a random element. Next, we simulate the corresponding term in line 6. with $\pi_1^* \leftarrow \mathsf{Sim}_1(\pi_1)$. In addition, there exists a simulator Sim'_1 for π'_1 such that the line 8. becomes $\pi_1^{**} \leftarrow \mathsf{Sim}'_1(\pi'_1)$.

Now suppose that there is a distinguisher between Game 2 and Game 3 with non-negligible advantage, then we show how to build an adversary $\mathcal{B}$ that breaks the CPA security of the FHE scheme. $\mathcal{B}$ has the following behaviour. $\mathcal{B}$ generates a public pk_0 using $\mathsf{FHE.Setup}(1^\lambda)$. It receives from the challenger of the CPA secure FHE scheme another public key pk_1. It runs $\mathcal{A}$ in order to get f_0, f_1, mpk and uses them for the CPA security game by choosing the messages (g_1, f_1). It forms $ct_0 := \mathsf{FHE.Enc}(\mathsf{pk}_0, f_0; R_0)$ (an encryption under his own key). Next, it receives an encryption $ct_1^\star$ of one of the two functions $\{g_1, f_1\}$ under pk_1 from the challenger. It can now use $\mathcal{A}$ in the following way by interacting as a legitimate

user. It simulates the first messages of $\mathcal{U}_0$ (line 5) with $(m_0 := (\mathsf{pk}_0, ct_{f_0}))$ and $\mathcal{U}_1$ with $(m_1 := (\mathsf{pk}_1, ct_1^\star))$. Up to this point, it could use the zero knowledge property and simulate the corresponding proofs π_1^* and π_1^{**}. Finally, $\mathcal{B}$ returns the same output of $\mathcal{A}$ (the same bit).

Now, taking a step back to the CPA security game, if $ct_1^\star$ is an encryption of f_1, then this situation corresponds to Game 2 experiment. If $ct_1^\star$ is an encryption of g_1 then it corresponds to Game 3 by construction. Unless the proofs are not zero-knowledge, the advantage of $\mathcal{B}$ winning the CPA security game of the FHE scheme is the same as the advantage of $\mathcal{A}$ in distinguishing between Game 2 and Game 3. We deduce that Game 2 is indistinguishable from Game 3, assuming that the FHE scheme is CPA-secure and the proofs π, π' are zero-knowledge.
Game 4. We change the behaviour of $\mathcal{U}_0$ as in the previous Game 3 by encrypting a randomly chosen function $g_0 \in F$ with size description equal to f_0 with a modified proof π_0 in the first message and π_0' in the second one. In more details, there exists a zero-knowledge simulator Sim_0 for π_0 that can simulate the proofs without knowing the underlying witness. We replace the corresponding term in line 3. for $\mathcal{U}_0$ with $ct_{g_0,0} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}_0, g_0; R_0)$, where R_0 is a random element. Next, we simulate the corresponding term in line 6. with $\pi_0^* \leftarrow \mathsf{Sim}_0(\pi_0)$ In addition, there exists a simulator Sim_0' for π_0' such that the line 8. becomes $\pi_0^{**} \leftarrow \mathsf{Sim}_0'(\pi_0')$. We can then proceed as for the transition between Game 2 and Game 3 to prove that Game 3 is indistinguishable from Game 4, assuming that the FHE scheme is CPA-secure and the proofs π, π' are zero-knowledge.

Putting all previous results together, we finally conclude that Game 0 is indistinguishable from Game 4. In Game 4, the view of $\mathcal{A}$ is independent of b: the functional keys $sk_{f_b}, sk_{f_{\bar{b}}}$ do not depend on the values sent by $\mathcal{A}$ by construction. Thus, the probability of guessing the bit b is exactly $1/2$. Hence, we conclude that this scheme satisfies the blindness property. □

4 Efficient Blind Interactive Inner Product FE

We want to stress here that for specific functionalities, our approach can lead to efficient constructions. We propose in this section a blind functional encryption for *inner product*, which is inspired by our generic construction. For such a functionality, we only need a *linearly* homomorphic encryption scheme, and for efficiency reasons, we chose to use CL scheme [16]. For most of the known IPFE scheme [2, 5], the functional key reduce to the computation of an inner product. The Setup algorithm consists of a description of a cyclic group $\mathbb{G}$ of prime order $p > 2^\lambda$ with generator $g \leftarrow \mathbb{G}$. For each $i \in \{1, \ldots, \ell\}$, it samples $s_i \leftarrow \mathbb{Z}_p$ and compute $h_i = g^{s_i}$. Finally, define $\mathsf{msk} := (s_i)_{i=1}^\ell$ and $\mathsf{mpk} := \left(\mathbb{G}, g, \{h_i\}_{i=1}^\ell\right)$. Note that the prime p is set for the CL scheme according to the IPFE. This is possible thanks to the flexibility of the CL key generation which is presented in App. B, and we refer to this appendix for notations.

Description of our scheme. The interactive key generation $\mathsf{IKeyGen}(\mathcal{AUT}(s \in \mathbb{Z}_p^\ell), \mathcal{U}(\mathsf{mpk}, y \in \mathbb{Z}_p^\ell))$, consisting of the two-party private inner product computation is as follows, which is an adaptation of [22].

- The user $\mathcal{U}$ generates a pair of keys $\mathsf{pk} = \mathfrak{g}_p^x$ and $\mathsf{sk} = x$ for the CL scheme over the message space $\mathbb{Z}_p$. Then, it encrypts each coordinate y_i for $i \in \{0, \ldots, \ell\}$ as $c_i = (c_{1,i}, c_{2,i}) = (\mathfrak{g}_p^{r_i}, \mathfrak{f}^{y_i}\mathfrak{h}^{r_i})$, sends pk, c_y to $\mathcal{AUT}$ and performs a ZKPoK $\pi_{\mathcal{U}}$ such that $\{\mathfrak{h} = \mathfrak{g}_p^x \wedge c_{1,i} = \mathfrak{g}_p^{r_i} \wedge c_{2,i} = \mathfrak{f}^{y_i}\mathfrak{h}^{r_i}$ for $i \in \{1, \ldots, \ell\}\}$.
- If the proof fails, $\mathcal{AUT}$ aborts. Otherwise, it homomorphically computes $c_{sk_y} := (c_{1,sk_y}, c_{2,sk_y}) \longleftarrow \left(\left(\prod_{i=1}^{\ell} c_{1,i}^{s_i}\right)\mathfrak{g}_p^{r'}, \left(\prod_{i=1}^{\ell} c_{2,i}^{s_i}\right)\mathfrak{h}^{r'}\right)$ for some random r' that it sends to $\mathcal{U}$. Then, it performs a proof $\pi_{\mathcal{AUT}}$ that: $\{\{g^{s_i} = h_i\}_{i=1}^{\ell} \wedge c_{sk_y} = \left(\left(\prod_{i=1}^{\ell} c_{1,i}^{s_i}\right)\mathfrak{g}^{r'}, \left(\prod_{i=1}^{\ell} c_{2,i}^{s_i}\right)\mathfrak{h}^{r'}\right)\}$.
- If $\pi_{\mathcal{AUT}}$ fails $\mathcal{U}$ aborts. It decrypts c_{sk_y} and gets $sk_y := (y, \langle s, y\rangle) \in \mathbb{Z}_p^{\ell} \times \mathbb{Z}_p$.

Our blindness notion is new in the context of FE, so it is difficult to find a point of comparison with existing classical 2PC protocols for computing inner-product, since they were not designed for this context. However, we could compare with other linearly homomorphic schemes. The additive variant of ElGamal would imply to compute a final discrete log, which is not possible for large p. The ZKPoK are proofs for classical discrete logarithm-based expressions. The main subtleties concern the CL part since it uses a group of unknown order, which can be obtained from class group of ideals of orders of imaginary quadratic fields. As in [14], the solution is to use repeated GPS proofs [21] with binary challenges to get special soundness. More efficient techniques have been recently proposed in [15]. For the proofs that concern the group $\mathbb{G}$ coming from the IPFE setup, a standard Schnorr proof is sufficient. Using Paillier encryption instead of CL prevents the necessity to repeat a GPS proof with binary challenge. It however necessitates to add (i) a proof that the Paillier modulus n has truly been computed as the multiplication of two primes p and q [12], (ii) a proof of knowledge of a plaintext y and its randomness r composing the given Paillier ciphertext $c = (1+n)^y r^n \pmod{n^2}$, which can be done using techniques given in [18] and (iii) a proof that $y < p$ in a group of composite order [9]. We argue that this implies a heavier global proof than what we propose using CL encryption.

Security and Efficiency Analysis of Our Inner Product IFE. The following result is a corollary of Thm. 6.

Theorem 7. *The scheme described above is message-private and blind if CL scheme is CPA-secure and the $\pi_{\mathcal{U}}, \pi_{\mathcal{AUT}}$ are zero-knowledge proofs of knowledge.*

A precise efficiency analysis of GPS-like proof in the context of CL encryption has been performed in [14]. It is implemented within class groups of some imaginary quadratic fields. The cost of such a proof is dominated by the computation of exponentiations in the class group. [14, Fig. 9] gives some measurements: on their architecture, an exponentiation takes 55ms for a 128 bit security. For the proof described in Eq. 4 with $\ell = 1$, there are essentially 4 exponentiations in the class group. This protocol has to be repeated say 40 times to get a soundness error of 2^{-40}, which means that such a proof costs less than 10 seconds (with $\ell = 1$). The overall cost is then linear in ℓ, which means that our interactive blind IFE has a reasonable practical cost of ℓ times tens of seconds. This is even more reasonable that this extraction is done only each time that a functional key is necessary, which happens occasionally.

Acknowledgements. The authors would like to thank Damien Stehlé for his suggestions during the redaction of this paper. All three authors were supported by the European Union H2020 Research and Innovation Program Grant 780701 (PROMETHEUS). The two first authors were also supported by the European Union H2020 Research and Innovation Program Grant 786767 (PAPAYA).

References

1. Abadi, M., Feigenbaum, J.: Secure circuit evaluation. Journal of Cryptology **2**(1), 1–12 (Feb 1990)
2. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg, Germany (2015)
3. Abe, M., Ohkubo, M.: A framework for universally composable non-committing blind signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 435–450. Springer, Heidelberg, Germany (2009)
4. Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: On the practical security of inner product functional encryption. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 777–798. Springer, Heidelberg, Germany (2015)
5. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg, Germany (2016)
6. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg, Germany (2010)
7. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: Hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg, Germany (2013)
8. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg, Germany (2011)
9. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg, Germany (2000)
10. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg, Germany (2015)
11. Camenisch, J., Kohlweiss, M., Rial, A., Sheedy, C.: Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 196–214. Springer, Heidelberg, Germany (2009)
12. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg, Germany (1999)
13. Canard, S., Diop, A., Kheir, N., Paindavoine, M., Sabt, M.: BlindIDS: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic.

In: Karri, R., Sinanoglu, O., Sadeghi, A.R., Yi, X. (eds.) ASIACCS 17. pp. 561–574. ACM Press (2017)

14. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from hash proof systems and efficient instantiations. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 191–221. Springer, Heidelberg, Germany (2019)
15. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold EC-DSA. In: PKC 2020, to appear (2020)
16. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 487–505. Springer, Heidelberg, Germany (2015)
17. Castagnos, G., Laguillaumie, F., Tucker, I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 733–764. Springer, Heidelberg, Germany (2018)
18. Cuvelier, E., Pereira, O., Peters, T.: Election verifiability or ballot privacy: Do we need to choose? In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 481–498. Springer, Heidelberg, Germany (2013)
19. Emura, K., Katsumata, S., Watanabe, Y.: Identity-based encryption with security against the KGC: A formal model and its instantiation from lattices. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019, Part II. LNCS, vol. 11736, pp. 113–133. Springer, Heidelberg, Germany (2019)
20. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 169–178. ACM Press (2009)
21. Girault, M., Poupard, G., Stern, J.: On the fly authentication and signature schemes based on groups of unknown order. Journal of Cryptology **19**(4), 463–487 (Oct 2006)
22. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: ICISC 2004. Lecture Notes in Computer Science, vol. 3506. Springer (2004)
23. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg, Germany (2007)
24. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg, Germany (2007)
25. Han, J., Susilo, W., Mu, Y., Zhou, J., Au, M.H.: PPDCP-ABE: Privacy-preserving decentralized ciphertext-policy attribute-based encryption. In: Kutylowski, M., Vaidya, J. (eds.) ESORICS 2014, Part II. LNCS, vol. 8713, pp. 73–90. Springer, Heidelberg, Germany (2014)
26. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures (extended abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg, Germany (1997)
27. Naveed, M., Agrawal, S., Prabhakaran, M., Wang, X., Ayday, E., Hubaux, J.P., Gunter, C.A.: Controlled functional encryption. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 1280–1291. ACM Press (2014)
28. Rial, A.: Blind attribute-based encryption and oblivious transfer with fine-grained access control. Des. Codes Cryptography **81**(2) (2016)

A Message-privacy for IFE.

Oracles. In traditional FE, the adversary has access to a $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle which extracts a functional key when the adversary requests it for a chosen input function f. We here adapt the definition of message-privacy to our interactive setting. The main difference relies in the fact that some information could leak during the interactive key generation. We introduce an *interactive* oracle $\mathsf{IKeyGen}(\mathcal{O}(\mathsf{msk}), \cdot)$: when calling this oracle, the adversary, on input $f \in F$, participates in an interactive protocol with the oracle playing the role of an honest authority. The adversary finally gets the output functional key sk_f. For any bit $b \in \{0, 1\}$, we define $\mathsf{Enc}_b(\mathsf{mpk}, \cdot, \cdot)$ to be an oracle which takes as inputs x_0 and x_1 and returns $\mathsf{Enc}(\mathsf{mpk}, x_b)$. The next definition extends known definitions [8] to the interactive setting and could be well adapted for private-key FE.

Definition 8 (Message-privacy). *Let* $\mathsf{IFE} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *over a message space* M *and a function space* F*. We say that* IFE *is* ***message-private*** *(MP) if for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that the quantity, called the advantage of* $\mathcal{A}$*,* $\mathsf{Adv}_{\mathcal{A},\mathit{MP\text{-}IFE}}(1^\lambda) := \left|\mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A}}^{(0),\mathsf{mp}}(\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A}}^{(1),\mathsf{mp}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{Exp}_{\mathcal{A}}^{(b),\mathsf{mp}}(\lambda)$ *is*

1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *2.* $b' \leftarrow \mathcal{A}^{\mathsf{IKeyGen}(\mathcal{O}(\mathsf{msk}),\cdot),\mathsf{Enc}_b(\mathsf{mpk},\cdot,\cdot)}(1^\lambda, \mathsf{mpk})$
3. output $b' = b$.

We required that for all $f \in F$ *and* (m_0, m_1) *coming from* $\mathcal{A}$*'s calls to the oracles* KeyGen *and* Enc_b *respectively, it holds that* $f(m_0) = f(m_1)$.

B The Castagnos-Laguillaumie scheme.

CL encryption scheme. The Setup phase in the CL scheme consists of the description of *a DDH group with an easy DL subgroup* $(p, \tilde{s}, \mathfrak{g}, \mathfrak{f}, \mathfrak{g}_p, G, F, G^p)$ where the set $(G, \cdot)$ is a cyclic group of order ps, for an unknown integer s, p is a prime number such that $\gcd(p, s) = 1$. The only known information on s is an upper bound $\tilde{s}$ of s. The set $G^p = \{\mathfrak{y}^p, \mathfrak{y} \in G\}$ is the subgroup of (unknown) order s of G, and F is the subgroup of order p of G, so that $G = F \times G^p$. The elements $\mathfrak{f}$, $\mathfrak{g}_p$ and $\mathfrak{g} = \mathfrak{f} \cdot \mathfrak{g}_p$ are respective generators of F, G^p and G. The discrete logarithm problem is easy in F, which means that there exists deterministic polynomial time algorithm a Solve that solves the discrete logarithm problem in F. The message space of CL is $\mathbb{Z}_p$ and its indistinguishability under chosen plaintext attacks relies on the hard subgroup membership assumption that says that is hard to distinguish the elements of G_p in G. An instantiation of this group is obtained using the class group of a non maximal order of an imaginary quadratic field (we refer the reader to [16, 17] for a more precise description). Roughly, CL scheme consists of a secret key sk is an integer $x \leftarrow \{0, \ldots, \tilde{s}p - 1\}$ and the public key is $\mathsf{pk} = \mathfrak{g}_p^x$. The encryption procedure returns a ciphertext $c_m = (c_1, c_2)$ where $c_1 \leftarrow \mathfrak{g}_p^r$ and $c_2 \leftarrow \mathfrak{f}^m \mathfrak{h}^r$ for a random r. The decryption algorithm computes $M \leftarrow c_2/c_1^x$ and returns m using the Solve algorithm on M.

C Leak-freeness

We provide a generalization of the Leak-Freeness property of [24].

Definition 9 (Leak-Freeness). *An* IKeyGen *protocol corresponding to* KeyGen *algorithm of any FE scheme is* leak-free *w.r.t.* KeyGen *if, for all efficient adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for all value λ, no distinguisher $\mathcal{D}$ can determine whether it is playing* GameReal *or* GameIdeal *where*

- GameReal*: Run* Setup(1^λ)*. As many times as $\mathcal{D}$ wants, $\mathcal{A}$ chooses a function f and executes the* IKeyGen$(\mathcal{AUT}, \cdot)$ *protocol input f with an honest authority $\mathcal{AUT}$. $\mathcal{A}$ returns the resulting view to $\mathcal{D}$ which returns a bit.*
- GameIdeal*: Run* Setup(1^λ)*. As many times as $\mathcal{D}$ wants, $\mathcal{S}$ chooses a function f and asks* Trivial.IKeyGen$(\mathsf{msk}, \cdot)$ *to obtain a functional key sk_f on input f. $\mathcal{S}$ returns then the resulting view to $\mathcal{D}$ which returns a bit.*

The quantity $\mathsf{Adv}_{\mathcal{D},\mathsf{leak-free}}(1^\lambda) := |\mathsf{Pr}[\mathcal{D}^{\mathsf{GameReal}}(1^\lambda) = 1] - \mathsf{Pr}[\mathcal{D}^{\mathsf{GameIdeal}}(1^\lambda) = 1]|$ *is the advantage of $\mathcal{D}$ and* IKeyGen *is* leak-free *w.r.t* KeyGen *if it is negligible.*

We discuss in the following some remarks about the definition.

- A secure two-party protocol realizing the KeyGen functionality of a classical FE ensures the message-privacy since it preserves each party for learning the other party's input. The main difference in our consideration is that we require the use of *a known* FE scheme with some specific KeyGen algorithm in addition to the existence of a simulator (which interacts with a specific oracle Trivial.IKeyGen). This simulator is then asked to produce a consistent view to any distinguisher. As mentioned in previous sections, a two-party protocol wouldn't offer the blindness property for free. In Example 2.1, Trivial.IKeyGen is by definition leak-free w.r.t KeyGen but not blind.
- The adversary in GameIdeal does not appear in the definition. As pointed in [24], the leak-freeness definition implies that the function (for the key being extracted) is *extractable* from the IKeyGen protocol (with all but negligible probability), since for every adversary it must exist a simulator $\mathcal{S}$ that should be able to interact with $\mathcal{A}$, in order to learn which functions to submit to the Trivial.IKeyGen$(\mathsf{msk}, \cdot)$ oracle.
- When considering the validity of sk_f (in Sec. 2.1), a ZKPoK is used in order to verify if a functional key sk_f is well-formed. This is independent from the definition of the leak-freeness property, since the authority is always honest in this context (simulated by an oracle).