# Practical Trade-Offs for Multi-Input Functional Encryption

Marc Joye[1] and Alain Passelègue[2]

[1] Technicolor (USA)
[2] ENS (France)

**Abstract.** Multi-input functional encryption is a paradigm that allows an authorized user to compute a certain function —and nothing more— over multiple plaintexts given only their encryption. The particular case of two-input functional encryption has very exciting applications like comparing the relative order of two plaintexts from their encrypted form, making range queries over an encrypted database, testing if two encrypted databases share common entries, and more.
While being extensively studied, multi-input functional encryption is not ready for a practical deployment, mainly for two reasons. First, known constructions rely on heavy cryptographic tools such as multilinear maps. Second, their security is still very uncertain, as revealed by recent devastating attacks.
This paper investigates a simpler approach. Rather than addressing multi-input functional encryption in its full generality, we target specific functions and relax the security notions. As a result, we obtain several practical realizations of multi-input encryption for specialized applications, including an efficient construction of order-revealing encryption with limited leakage, under the standard DLin assumption.

**Keywords:** Multi-input functional encryption; order-revealing encryption; property-preserving encryption; encrypted databases; cloud-based services.

## 1 Introduction

The growing reliance on numerous cloud-based services for storing and processing sensitive data demonstrated limitations of traditional encryption techniques. Specifically, traditional encryption is an all-or-nothing notion: informally, an unauthorized user (i.e., who has not access to the decryption key) should not learn any information whatsoever about a plaintext given its encryption. But in many use cases, there is often a need to get a much more fine-grained control of the decryption policy.

(Multi-Input) Functional Encryption. The paradigm of *functional encryption* [8,20] is an extension of traditional encryption that enables an authorized user to compute a certain function of the plaintext. Each decryption key $\mathsf{sk}_f$ corresponds to a specific function $f$. Informally, this private key $\mathsf{sk}_f$, given the encryption of a plaintext $x$, allows her holder to learn $f(x)$, and nothing more. An important subclass of functional encryption is *predicate encryption* [9,16]. A plaintext $x$ is viewed as pair $(I, \dot{x})$ where $I$ is some attribute (associated to the message) and $\dot{x}$ is the message itself; functionality $f$ is then defined as

$$f(I, \dot{x}) = \begin{cases} \dot{x} & \text{if } P(I) = 1, \text{ and} \\ \perp & \text{otherwise} \end{cases}$$

for a given predicate $P$.

The function can be defined over *multiple* plaintexts given their corresponding ciphertexts. This gives rise to multi-input functional encryption [7,15]. Of particular interest is the case of two-input functional encryption. Suppose that given two encrypted plaintexts, a cloud-based service wishes to compute their respective ordering. For a *public* comparison function, such a functionality is offered by *order-revealing encryption* (ORE) [5,7]. We note that order-revealing encryption necessarily requires *secret-key* encryption as otherwise a binary search from the encryption of chosen plaintexts would yield bit-by-bit the decryption of a given target ciphertext using the ORE comparison procedure. ORE can thus be seen as a secret-key two-input functional encryption for (public) comparison. It is a very useful primitive as it allows to answer queries over encrypted data, including range queries, sorting queries, searching queries, and more [1,4].

From OPE to ORE. Order-revealing encryption evolved from *order-preserving encryption* (OPE) [4,5], an encryption primitive that preserves the relative ordering of the plaintexts. Clearly, an OPE scheme cannot achieve the standard security notion of *indistinguishability under chosen-plaintext attacks* (IND-CPA). The best we can hope from an OPE scheme is that the encryption of a sequence of plaintexts reveals nothing beyond their relative ordering, the resulting security notion is termed IND-OCPA. Unfortunately, Boldyreva *et al.* showed in [4] that it is impossible to efficiently meet this natural security notion of IND-OCPA, even when the size of the ciphertext space is exponentially larger than that of the message space.

The situation for ORE schemes is different. In [7], Boneh *et al.* present an ORE scheme actually meeting the analogue of IND-OCPA security. But their construction is mostly of existential nature and as such should be considered as a possibility result. The candidate ORE scheme presented in [7] is hardly implementable since it relies on heavy cryptographic tools, namely $(\ell/2 + 1)$-way multilinear maps for comparing $\ell$-bit values. Furthermore, and maybe more importantly, the underlying security assumption is questionable owing to the recent attacks mounted against multilinear maps [12,13].

ORE in Practice. A practical construction for order-revealing encryption is proposed in [11]. It merely requires a pseudorandom function $F$ with output space $\{0, 1, 2\}$. The encryption under secret key $K$ of an $\ell$-bit plaintext $x = m_1 m_2 \cdots m_\ell$ with $m_i \in \{0, 1\}$, $\mathsf{ct} = (c_1, c_2, \ldots, c_\ell)$, is obtained iteratively as

$$c_i = F\big(K, (i, m_1 m_2 \cdots m_{i-1} \| 0^{\ell-i})\big) + m_i \pmod 3, \quad \text{for } 1 \leq i \leq \ell \ .$$

The comparison of two ciphertexts $\mathsf{ct} = (c_1, c_2, \ldots, c_\ell)$ and $\mathsf{ct}' = (c'_1, c'_2, \ldots, c'_\ell)$, corresponding to plaintexts $x$ and $x'$, is conducted by finding the first index $i$, $1 \leq i \leq \ell$, such that $c'_i \neq c_i$. Then,

$$\begin{cases} x < x' & \text{if there exists such an index } i \text{ and if } c'_i = c_i + 1 \pmod 3 \\ x \geq x' & \text{otherwise} \end{cases} .$$

While this construction is very efficient, it has the drawback of leaking an important amount of information, as one obtains immediately, given two ciphertexts, the size of the largest common prefix of the two corresponding plaintexts. In particular, this provides an upper bound on the distance separating the two plaintexts.

Our contributions. In this paper, we investigate a new approach towards building efficient secret-key multi-input functional encryption. We propose the notion of dedicated multi-input functional encryption, which can be viewed both as a generalization of the notion of property-preserving encryption [10,19] and as a specialization of the notion of multi-input functional encryption. Basically, a dedicated multi-input functional encryption scheme is a secret-key encryption scheme associated to a $k$-ary function $f$. The encryption algorithm takes as input a secret key, a message, and some index $i \in [k]$ and outputs a ciphertext. Moreover, there exists a public procedure such that, given $k$ ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, each corresponding to an encryption of a message $x_i$ at index $i$, for $i \in [k]$, one can compute $f(x_1, \ldots, x_k)$.

We consider two (indistinguishability-based and simulation-based) security notions that take into account a possible leakage. The leakage comprises at least the information resulting from the evaluation function, which is unavoidable. However, contrary to a perfect solution that would only permit this unavoidable leakage (as the one offered in [7]), we allow for additional leakage, provided it is very limited. Doing so, we are able to devise constructions that can be used in practical applications. Our main construction is an efficient order-revealing encryption scheme with limited leakage, under standard assumptions.

Of independent interest, we also provide a very simple construction achieving the best possible security for short messages.

## 2 Definitions

### 2.1 Dedicated Multi-Input Functional Encryption

We introduce the paradigm of dedicated multi-input functional encryption (DMIFE), as a generalization of property-preserving encryption defined by Pandey and Rouselakis [19] as well as a weakening of the general notion of multi-input functional encryption [7,15]. Our notion assumes the private-key setting [21] and corresponds to *specialized* multi-input functional encryption schemes where the evaluation of the function is public (i.e., no functional secret key is involved).

**Definition 1 (Dedicated Multi-Input Functional Encryption).** *A* dedicated multi-input functional encryption scheme *for a $k$-ary function $f$ consists of a tuple of algorithms $\mathcal{DMIFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$, defined below.*

- $\mathsf{Setup}(1^\kappa)$ *is a probabilistic algorithm that takes as input the security parameter $1^\kappa$ and outputs a secret key* $\mathsf{sk}$ *(and public parameters* $\mathsf{pp}$ *—including the message space $\mathcal{M}$).*
- $\mathsf{Enc}(i, \mathsf{sk}, x)$ *takes as input an index $i \in [k]$, a key* $\mathsf{sk}$*, and a message $x \in \mathcal{M}$. It outputs a ciphertext* $\mathsf{ct}$*.*
  *Index $i$ indicates that the output ciphertext* $\mathsf{ct}$ *constitutes the $i$-th input to function $f$.*
- $\mathsf{Eval}_f(\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ *takes as input $k$ ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ and outputs a value $y$ in the range of $f$.*

*For correctness, it is required that for all $\mathsf{sk} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$ and all $(x_1, \ldots, x_k) \in \mathcal{M}^k$:*

$$\mathsf{Eval}_f(\mathsf{ct}_1, \ldots, \mathsf{ct}_k) = f(x_1, \ldots, x_k) \quad \textit{where } \mathsf{ct}_i = \mathsf{Enc}(i, \mathsf{sk}, x_i) \ .$$

*Remark 2.* 1. Definition 1 is "asymmetric" in the sense that a given ciphertext is bound to a specific input position in the $\mathsf{Eval}_f$ procedure. We could define a "symmetric" version of dedicated multi-input functional encryption where the encryption algorithm $\mathsf{Enc}$ no longer takes in an index $i \in [k]$ so that a ciphertext can be used in any input position for the $\mathsf{Eval}_f$ procedure. We do not study this symmetric version further since, as stated in Lemma 5, it is implied by the asymmetric version.
2. We choose not to include a decryption algorithm in our definition, since this omission is without loss of generality. Indeed, if necessary, one could just augment the encryption of a message $x$ with an encryption of $x$ with a CPA-secure symmetric encryption scheme under a specific secret-key. Via CPA-security, this additional information does not compromise the security of the construction.
3. In this paper, we focus on the three following functions:
   - $f_\perp \colon\ (\boldsymbol{a}, \boldsymbol{b}) \mapsto \begin{cases} 1 & \text{if } \langle \boldsymbol{a}, \boldsymbol{b} \rangle = 0 \\ 0 & \text{otherwise} \end{cases}$ ;
   - $f_\# \colon\ (\mathcal{S}, \mathcal{T}) \mapsto \#(\mathcal{S} \cap \mathcal{T})$ ;
   - $f_< \colon\ (x, y) \mapsto \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$ .

### 2.2 Two Security Flavors

We examine two different security notions and explore the relations between them. The first notion is defined as an indistinguishability-based security game, while the second (and stronger) one as a simulation-based security game. These are generalizations of classical notions considered in the case of property-preserving encryption, e.g. in [3,10,14,19].

The two notions are defined relatively to a leakage function $\mathcal{L}$. As a DMIFE scheme for a function $f$ has to reveal, via the $\mathsf{Eval}_f$ procedure, at least the values of the function $f$ according to any tuple of $k$ messages $x_1, \ldots, x_k$ such that $x_i$ is encrypted for index $i \in [k]$, $\mathcal{L}$ will contain at least this information This leakage is written $\mathcal{L}_f$ and is defined below.

**Definition 3 (Leakage of a Function).** *The* leakage $\mathcal{L}_f$ *of a* $k$*-ary function* $f$ *with respect to* $k$ *vectors* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ *of* $q_1, \ldots, q_k$ *messages respectively —one vector of messages per position in the input of the function, so* $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,q_i})$*— is defined as:*

$$\mathcal{L}_f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) = (f(x_{1,i_1}, \ldots, x_{k,i_k}))_{i_1 \in [q_1], \ldots, i_k \in [q_k]} \ .$$

$\mathcal{L}$-Indistinguishability Security. A DMIFE scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$ for a $k$-ary function $f$ is $\mathcal{L}$-indistinguishability secure if, for any two sequences of plaintexts with the same leakage, the corresponding sequences of ciphertexts are computationally indistinguishable. Security is defined by a variant of the standard semantic security game and is depicted in Figure 1.

Specifically, the adversary has black-box access to a left-or-right encryption oracle **LoR**. This oracle can be adaptively queried with an index $i$ and a pair of messages $(x^{(0)}, x^{(1)})$ to get $\mathsf{Enc}(i, \mathsf{sk}, x^{(b)})$ with $b$ being a fixed bit and $\mathsf{sk}$ being a secret key, initialized by the **Initialize** procedure. At the end, the adversary outputs a bit $b'$ and wins if $b = b'$; namely, $\mathbf{Finalize}(b') = 1$. In order to prevent trivial attacks (i.e., attacks resulting from the leakage function), the adversary is restricted as follows. If $((x_{i,1}^{(0)}, x_{i,1}^{(1)}), \ldots, (x_{i,q_i}^{(0)}, x_{i,q_i}^{(1)}))$ denotes the sequence of $q_i$ queries made with index $i$ to the **LoR** oracle then, letting $\boldsymbol{x}_i^{(t)} = (x_{i,1}^{(t)}, \ldots, x_{i,q_i}^{(t)})$ for $t \in \{0,1\}$, the sequence of queries made by the adversary has to satisfy:

$$\mathcal{L}(\boldsymbol{x}_1^{(0)}, \ldots, \boldsymbol{x}_k^{(0)}) = \mathcal{L}(\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_k^{(1)}) \ .$$

```
proc Initialize
$b \leftarrow \{0,1\}$
$\mathsf{sk} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$
For $i \in [k]$:
    $\boldsymbol{\ell}_i^{(0)}, \boldsymbol{\ell}_i^{(1)} \leftarrow ()$

proc Finalize($b'$)
Return $b' = b$

proc LoR($i, x^{(0)}, x^{(1)}$)
$\boldsymbol{\ell}_i^{(0)} \leftarrow \boldsymbol{\ell}_i^{(0)}.\mathsf{append}(x^{(0)})$
$\boldsymbol{\ell}_i^{(1)} \leftarrow \boldsymbol{\ell}_i^{(1)}.\mathsf{append}(x^{(1)})$
If $\mathcal{L}(\boldsymbol{\ell}_1^{(0)}, \ldots, \boldsymbol{\ell}_k^{(0)}) \neq \mathcal{L}(\boldsymbol{\ell}_1^{(1)}, \ldots, \boldsymbol{\ell}_k^{(1)})$:
    Return $\perp$
Else:
    $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(i, \mathsf{sk}, x^{(b)})$
    Return $\mathsf{ct}$
```

**Fig. 1.** Game defining the $\mathcal{L}$-indistinguishability security of a DMIFE scheme.

$\mathcal{L}$-Simulation Security. A DMIFE scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$ for a $k$-ary function $f$ is $\mathcal{L}$-simulation secure if, for any efficient adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_q)$ which is given black-box access to encryption oracle $\mathsf{Enc}$ that it queries $q$ times, there exists an efficient stateful simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q)$ such that the outputs of the two distributions $\mathsf{Real}^{\mathcal{DMIFE}}_{\mathcal{A}}(\kappa)$ and $\mathsf{Sim}^{\mathcal{DMIFE}}_{\mathcal{A},\mathcal{S},\mathcal{L}}(\kappa)$, described in Figure 2, are computationally indistinguishable.

```
proc $\mathsf{Real}^{\mathcal{DMIFE}}_{\mathcal{A}}(\kappa)$
$\mathsf{sk} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$
$\mathsf{st}_\mathcal{A} \leftarrow \mathcal{A}_0(1^\kappa)$
For $i \in [k]$:
    $\mathbf{ct}_i \leftarrow ()$
For $C \in [q]$:
    $((i, x), \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}_C(\mathsf{st}_\mathcal{A}, (\mathbf{ct}_1, \ldots, \mathbf{ct}_k))$
    $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(i, \mathsf{sk}, x)$
    $\mathbf{ct}_i \leftarrow \mathbf{ct}_i.\mathsf{append}(\mathsf{ct})$
Return $(\mathbf{ct}_1, \ldots, \mathbf{ct}_k)$
```

```
proc $\mathsf{Sim}^{\mathcal{DMIFE}}_{\mathcal{A},\mathcal{S},\mathcal{L}}(\kappa)$
$\mathsf{st}_\mathcal{S} \leftarrow \mathcal{S}_0(1^\kappa)$
$\mathsf{st}_\mathcal{A} \leftarrow \mathcal{A}_0(1^\kappa)$
For $i \in [k]$:
    $\mathbf{ct}_i \leftarrow ()$
For $C \in [q]$:
    $((i, x), \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}_C(\mathsf{st}_\mathcal{A}, (\mathbf{ct}_1, \ldots, \mathbf{ct}_k))$
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i.\mathsf{append}(x)$
    $(\mathsf{ct}, \mathsf{st}_\mathcal{S}) \xleftarrow{\$} \mathcal{S}_C(\mathsf{st}_\mathcal{S}, \mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k))$
    $\mathbf{ct}_i \leftarrow \mathbf{ct}_i.\mathsf{append}(\mathsf{ct})$
Return $(\mathbf{ct}_1, \ldots, \mathbf{ct}_k)$
```

**Fig. 2.** Game defining the $\mathcal{L}$-simulation security of a DMIFE scheme.

## 2.3 Relations Between These Security Notions

As one could expect, simulation security implies indistinguishability security, as stated in the following lemma. Moreover, as already mentioned in Remark 2, for both security notions, the existence of a secure "asymmetric" DMIFE implies the existence of secure "symmetric" DMIFE, as stated in Lemma 5.

**Lemma 4.** *Assuming $\mathcal{DMIFE}$ is an $\mathcal{L}$-simulation secure dedicated multi-input functional encryption scheme, then $\mathcal{DMIFE}$ is an $\mathcal{L}$-indistinguishability secure dedicated multi-input functional encryption scheme.*

**Lemma 5.** *Assuming there exists an $\mathcal{L}$-indistinguishability (resp. $\mathcal{L}$-simulation) secure asymmetric dedicated multi-input functional encryption scheme for a function $f$, there exists a $\mathsf{sym}_\mathcal{L}$-indistinguishability (resp. $\mathsf{sym}_\mathcal{L}$-simulation) secure symmetric dedicated multi-input functional encryption scheme for the function $f$, with $\mathsf{sym}_\mathcal{L}(\boldsymbol{x}) = \mathcal{L}(\boldsymbol{x}, \ldots, \boldsymbol{x})$.*

The proofs of these two lemmata are detailed in Appendix A.

## 3 Order-Revealing Encryption with Simulation-Security for Polynomial-Size Message Space

Before starting to build our main construction, we would like to remark that, while it seems extremely hard to obtain an $\mathcal{L}_{f_<}$-indistinguishability secure dedicated

2-input functional encryption for the function $f_<$, also called order-revealing encryption in the "symmetric" case, from standard assumptions, there is actually a very simple construction that even achieves simulation-based security assuming only one-way functions, for polynomial-size message space. To improve efficiency, our construction can be instantiated using a pseudorandom permutation, such as AES. This leads to a very efficient construction for small message spaces (*e.g.*, 10-bit integers).

Let $\{0,\ldots,N-1\}$ denote the message space, and let $F\colon\ \{0,1\}^\kappa \times \mathcal{D} \to \mathcal{R}$ be a pseudorandom function such that its domain $\mathcal{D}$ contains $\{0,\ldots,N-1\} \times \{0,\ldots,2N-1\}$.

**Construction 1.** We define $\mathcal{D2IFE}_< = (\mathsf{Setup}_<, \mathsf{Enc}_<, \mathsf{Eval}_{f_<})$ as follows:

– $\mathsf{Setup}_<(1^\kappa)$ picks $K \overset{\$}{\leftarrow} \{0,1\}^\kappa$ at random and returns it as the secret key $\mathsf{sk}$;
– $\mathsf{Enc}_<(i,\mathsf{sk},x)$ is defined as:

$$\mathsf{Enc}_<(i,K,x) = \begin{cases} \mathsf{shuffle}(F_K(x,x+1),\ldots,F_K(x,x+N-1)) & \text{if } i=1 \\ \mathsf{shuffle}(F_K(0,x),\ldots,F_K(N-1,x)) & \text{if } i=2 \end{cases};$$

[Here $\mathsf{shuffle}$ is a randomized algorithm that returns a random shuffling of its inputs.]
– $\mathsf{Eval}_{f_<}(\mathsf{ct}_1,\mathsf{ct}_2)$ checks whether there is a common value in $\mathsf{ct}_1$ and $\mathsf{ct}_2$. If so, it outputs 1; if not, it outputs 0.

Correctness. It is clear that if there is no common value, the output of the evaluation algorithm, "$\geq$", is correct. However, it might happen that there is a common value due to a collision. Hence, to ensure that this does not happen, we might want $F_K$ to be injective (e.g., using a pseudorandom permutation instead of a pseudorandom function), but one could simply make the range $\mathcal{R}$ big enough so that the probability of a collision is negligible.

Construction 1 being deterministic, it reveals if two ciphertexts encrypted with the same index corresponds to the same plaintext. This is the only extra information, beyond the relative order, that is leaked. However, this extra-information is always leaked in the "symmetric" case, as one can always check, given two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$ corresponding to plaintexts $x_1, x_2$, whether $x_1 \geq x_2$ and $x_2 \geq x_1$. Thus, if $x_1 = x_2$, the equality is revealed. For this reason, we claim that Construction 1 achieves ideal security, and we define its leakage $\mathcal{L}_{<,=}$ as:

$$\mathcal{L}_{<,=}(\boldsymbol{x}_1,\boldsymbol{x}_2) = (\mathcal{L}_{f_<}(\boldsymbol{x}_1,\boldsymbol{x}_2), \mathcal{L}_=(\boldsymbol{x}_1,\boldsymbol{x}_2))\,,$$

with $\mathcal{L}_=(\boldsymbol{x}_1,\boldsymbol{x}_2) = (\mathbb{1}_=(x_{b,i_b}, x_{b,j_b}))_{i_b,j_b \in [|\boldsymbol{x}_b|], b\in\{1,2\}}$ where $\mathbb{1}_=(a,b)$ returns 1 if and only if $a=b$. Precisely, $\mathcal{L}_{f_<}(\boldsymbol{x}_1,\boldsymbol{x}_2)$ reveals exactly the relative order of messages encrypted with index 1 relatively to messages encrypted with index 2, while $\mathcal{L}_=(\boldsymbol{x}_1,\boldsymbol{x}_2)$ reveals exactly the pairs of equal messages encrypted with the same index.

**Theorem 6.** *Assuming one-way functions exist, there exists an $\mathcal{L}_{<,=}$-simulation secure dedicated 2-input functional encryption scheme for the function $f_<$, for polynomial-size message spaces.*

The proof of the above theorem is detailed in Appendix B.

## 4 Orthogonality Testing and Relation with Predicate-Encryption

In this section, we describe how we obtain a dedicated 2-input functional encryption scheme for orthogonality testing, namely for the function

$$f_\perp \colon \ (\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_p^n \mapsto \begin{cases} 1 & \text{if } \langle \boldsymbol{a}, \boldsymbol{b} \rangle = 0 \\ 0 & \text{otherwise} \end{cases} .$$

This is a first step in building our efficient order-revealing encryption scheme with limited leakage.

The existence of such a scheme is immediately implied by the existence of a fully-secure secret-key inner-product encryption scheme, which in particular exists under the DLin assumption [6], as proven in [18]. More generally, we describe a transformation from any fully-secure secret-key predicate encryption for a class of predicate $\mathcal{F}_f = \{f_a \colon b \in \mathcal{M} \mapsto f(a,b) \in \{0,1\} \mid a \in \mathcal{M}\}$ to a dedicated 2-input functional encryption scheme for the function $f$. A very similar result was already proposed in the case of property-preserving encryption in [2,10]. For completeness, definitions of the DLin assumption and of fully-secure secret-key predicate encryption and inner-product encryption are recalled in Appendix C. Please note, in particular, that by fully-secure, we mean predicate-hiding and attribute-hiding.

**Theorem 7.** *Let $f \colon \mathcal{M} \times \mathcal{M} \to \{0,1\}$ be any function. Assuming there exists a fully-secure secret-key predicate encryption scheme for the class of predicates $\mathcal{F}_f = \{f_a \colon b \in \mathcal{M} \mapsto f(a,b) \in \{0,1\} \mid a \in \mathcal{M}\}$, then there exists an $\mathcal{L}_f$-indistinguishability secure dedicated 2-input functional encryption scheme for the function $f$.*

*Proof.* Let $(\mathsf{Setup}, \mathsf{TokenGen}, \mathsf{Enc}, \mathsf{Dec})$ be a fully-secure secret-key predicate encryption scheme for the class of predicates $\mathcal{F}_f$. We build an $\mathcal{L}_f$-indistinguishability secure dedicated 2-input functional encryption scheme $(\mathsf{Setup}_f, \mathsf{Enc}_f, \mathsf{Eval}_f)$ for $f$ as follows: $\mathsf{Setup}_f$ is the same as $\mathsf{Setup}$. $\mathsf{Enc}_f(i, \mathsf{sk}, x)$ returns $\mathsf{TokenGen}(\mathsf{sk}, x)$ if $i = 1$ and $\mathsf{Enc}(\mathsf{sk}, (x, 1))$ if $i = 2$, meaning that it encrypts 1 with the attribute $x$. Finally, $\mathsf{Eval}_f(\mathsf{ct}_1, \mathsf{ct}_2)$ simply uses $\mathsf{ct}_1$ to decrypt $\mathsf{ct}_2$, and return 1 if and only if the decryption outputs 1, and 0 otherwise. Both correctness and security immediately follow from the correctness and the security of the underlying predicate encryption scheme, and Theorem 7 follows. □

Let $\mathcal{L}_\perp$ denote the leakage of the function $f_\perp$ (so $\mathcal{L}_\perp = \mathcal{L}_{f_\perp}$ according to Definition 3), so $\mathcal{L}_\perp$ reveals exactly all the pairs of orthogonal vectors $(\boldsymbol{a}, \boldsymbol{b})$ such that $\boldsymbol{a}$ is encrypted at index 1 and $\boldsymbol{b}$ is encrypted at index 2.

**Corollary 8.** *Assuming* DLin*, there exists an $\mathcal{L}_\perp$-indistinguishability secure 2-input functional encryption scheme for orthogonality testing.*

*Proof.* Corollary 8 follows immediately from the existence of a fully-secure inner-product encryption scheme under the DLin assumption, as proven in [18], and from Theorem 7. □

For the rest of the paper, we denote by $\mathscr{D2IFE}_\perp = (\mathsf{Setup}_\perp, \mathsf{Enc}_\perp, \mathsf{Eval}_{f_\perp})$ an $\mathcal{L}_\perp$-indistinguishability secure dedicated 2-input functional encryption scheme for orthogonality testing.

## 5 Computing Cardinality of Intersection with Limited Leakage

We now describe the second step in building our efficient order-revealing encryption scheme with limited leakage, which is to build a dedicated 2-input functional encryption scheme for computing the cardinality of intersection. Specifically, the messages are sets of fixed size $n$ and the function $f$ we target is the function $f_\#\colon (\mathcal{S}_1, \mathcal{S}_2) \mapsto \#(\mathcal{S}_1 \cap \mathcal{S}_2)$. Our construction relies on the existence of a dedicated 2-input functional encryption scheme for $f_\perp$.

In order to ease the reading, we assume that every set in the message space has a fixed size $n$. One could circumvent this condition as long as the maximal size of a set is known and fixed in advance, but this is not useful for our purpose.

We compute the cardinality of the intersection of two sets as follows: given two sets of integers $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\mathcal{B} = \{b_1, \ldots, b_n\}$, one can compute the polynomial $P_\mathcal{A}(X) = \prod_{i=1}^n (X - a_i)$ such that $b \in \mathcal{A} \Leftrightarrow P_\mathcal{A}(b) = 0$. The problem is that this technique does not hide anything about elements in $\mathcal{A}$ and $\mathcal{B}$. To address this issue, one simply notices that, given $P_\mathcal{A}(X) = \sum_{i=0}^n \alpha_i \cdot X^i$, testing $P_\mathcal{A}(b) = 0$ simply consists in checking if $\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle = 0$, with $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_n)$ and $\boldsymbol{\beta} = (1, b, b^2, \ldots, b^n)$. Therefore, this can be tested privately using a dedicated 2-input functional encryption for orthogonality testing.

We denote by $\mathsf{coef}(\mathcal{S})$ the vector $(\alpha_0, \ldots, \alpha_n)$ such that $\prod_{s \in \mathcal{S}} (X - s) = \sum_{i=0}^n \alpha_i \cdot X^i$ and by $\mathsf{exp}(s)$ the vector $(1, s, s^2, \ldots, s^n)$. It is straightforward that, for $n$ being polynomial, computations of $\mathsf{coef}(\mathcal{S})$ and $\mathsf{exp}(s)$ are polynomial-time. Let $\mathscr{D2IFE}_\perp = (\mathsf{Setup}_\perp, \mathsf{Enc}_\perp, \mathsf{Eval}_{f_\perp})$ be a dedicated 2-input functional encryption scheme for orthogonality testing.

**Construction 2.** We build a dedicated 2-input functional encryption scheme $\mathscr{D2IFE}_\# = (\mathsf{Setup}_\#, \mathsf{Enc}_\#, \mathsf{Eval}_{f_\#})$ for the function $f_\#$ as follows:

- $\mathsf{Setup}_\#$ takes as input the security parameter $\kappa$ and outputs $\mathsf{Setup}_\perp(1^\kappa) = \mathsf{sk}$;
- $\mathsf{Enc}_\#$ takes as input an index $i \in \{1, 2\}$, a secret key $\mathsf{sk}$, and a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ and outputs:

$$\mathsf{Enc}_\#(i, \mathsf{sk}, \mathcal{S}) = \begin{cases} \mathsf{Enc}_\perp(1, \mathsf{sk}, \mathsf{coef}(\mathcal{S})) & \text{if } i = 1\,; \\ \mathsf{shuffle}(\mathsf{Enc}_\perp(2, \mathsf{sk}, \mathsf{exp}(s_1)), \ldots, & \\ \qquad\qquad \mathsf{Enc}_\perp(2, \mathsf{sk}, \mathsf{exp}(s_n))) & \text{if } i = 2\,. \end{cases}$$

– $\mathsf{Eval}_{f_{\#}}$ takes as input a pair of ciphertexts $(\mathsf{ct}_1, \mathsf{ct}_2)$ encrypted with index 1 and 2 respectively and with $\mathsf{ct}_2 = (\mathsf{ct}_{2,1}, \ldots, \mathsf{ct}_{2,n})$, computes $y_i = \mathsf{Eval}_{f_{\perp}}(\mathsf{ct}_1, \mathsf{ct}_{2,i})$ for $i = 1, \ldots, n$ and outputs $\sum_{i=1}^{n} y_i$.

Correctness. Correctness follows immediately from the correctness of $\mathcal{D2IFE}_{\perp}$.

Security. To compute the size of the intersection of a set $\mathcal{S}$ encrypted with index 1 with a set $\mathcal{T}$ encrypted with index 2, one checks, for every element $t \in \mathcal{T}$, if $t \in \mathcal{S}$. Therefore, while it clearly allows to compute the size of the intersection, this also leaks more information. Indeed, consider two sets $\mathcal{S}_1$ and $\mathcal{S}_2$ encrypted with index 1 and another set $\mathcal{T}$ encrypted with index 2. Then, for every $t \in \mathcal{T}$, one can check if $t \in \mathcal{S}_1$ and if $t \in \mathcal{S}_2$. Hence, not only the cardinality $\mathcal{T} \cap \mathcal{S}_1$ and $\mathcal{T} \cap \mathcal{S}_2$ is revealed, but also the one of $\mathcal{T} \cap \mathcal{S}_1 \cap \mathcal{S}_2$. More generally, if $k$ sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ are encrypted with index 1 and a set $\mathcal{T}$ is encrypted with index 2, their encryptions reveal the size of the intersection of $\mathcal{T}$ with any intersection of 1 to $k$ different sets from $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$.

We prove that this is exactly the information that is leaked by our construction and define the leakage of our construction, denoted $\mathcal{L}_{\#^*}$, as follows. For two sequences of sets $\boldsymbol{\mathcal{S}} = (\mathcal{S}_1, \ldots, \mathcal{S}_{q_1})$ and $\boldsymbol{\mathcal{T}} = (\mathcal{T}_1, \ldots, \mathcal{T}_{q_2})$ encrypted respectively with index 1 and 2, we define:

$$\mathcal{L}_{\#^*}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}}) = (\#(\mathcal{I} \cap \mathcal{T}_i))_{\mathcal{I} \in \boldsymbol{\mathcal{S}}^{\cap}, i \in [q_2]} \ ,$$

where $\boldsymbol{\mathcal{S}}^{\cap} = \{\mathcal{S}_{i_1} \cap \cdots \cap \mathcal{S}_{i_j} \mid j \in [q_1], i_j \in [q_1]\}$, so $\boldsymbol{\mathcal{S}}^{\cap}$ contains every intersection of 1 to $q_1$ different sets encrypted at index 1. In particular, every set $\mathcal{S}_i$ is in $\boldsymbol{\mathcal{S}}^{\cap}$.

**Theorem 9.** *Assuming there exists an $\mathcal{L}_{\perp}$-indistinguishability secure dedicated 2-input functional encryption scheme for orthogonality testing, there exists an $\mathcal{L}_{\#^*}$-indistinguishability secure dedicated 2-input functional encryption scheme for cardinality of intersection.*

*Remark 10.* Note that, even if $\mathcal{L}_{\#^*}$ is formally an exponential-size vector, checking whether a query made by an adversary is valid or not remains polynomial. Indeed, for every new query $(1, \mathcal{S}^{(0)}, \mathcal{S}^{(1)})$, one just needs to check that for every element $t^{(0)}$ in a set $\mathcal{T}^{(0)}$ corresponding to a previous query $(2, \mathcal{T}^{(0)}, \mathcal{T}^{(1)})$ such that $t^{(0)}$ is also in sets $\mathcal{S}^{(0)}_{i_1}, \ldots, \mathcal{S}^{(0)}_{i_j}$ corresponding to previous queries $(1, \mathcal{S}^{(0)}_{i_1}, \mathcal{S}^{(1)}_{i_1}), \ldots, (1, \mathcal{S}^{(0)}_{i_j}, \mathcal{S}^{(1)}_{i_j})$, there is also an element in $t^{(1)}$ in the set $\mathcal{T}^{(1)}$ such that $t^{(1)}$ is also in the sets $\mathcal{S}^{(1)}_{i_1}, \ldots, \mathcal{S}^{(1)}_{i_j}$, which is clearly a polynomial-time process. A similar process can be done for every new query made with index 2. Therefore, the security game defining $\mathcal{L}_{\#^*}$-indistinguishability security remains polynomial-time.

*Proof (Theorem 9).* Let $\mathcal{A}$ be an adversary against the $\mathcal{L}_{\#^*}$-indistinguishability security of the scheme $\mathcal{D2IFE}_{\#} = (\mathsf{Setup}_{\#}, \mathsf{Enc}_{\#}, \mathsf{Eval}_{f_{\#}})$ obtained via Construction 2, that makes $q_1$ queries with index 1 and $q_2$ queries with index 2 to its encryption oracle. Then one can design an adversary $\mathcal{B}$ against the $\mathcal{L}_{\perp}$-indistinguishability security of $\mathcal{D2IFE}_{\perp} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_{f_{\perp}})$ as follows: $\mathcal{B}$

starts by initializing two empty lists $\mathsf{list}_0, \mathsf{list}_1$. Next, $\mathcal{B}$ runs adversary $\mathcal{A}$. When the latter makes a query $(i, \mathcal{S}^{(0)}, \mathcal{S}^{(1)})$, $\mathcal{B}$ does the following.

If $i = 1$, $\mathcal{B}$ first adds $\mathcal{S}^{(0)}$ to $\mathsf{list}_0$ and $\mathcal{S}^{(1)}$ to $\mathsf{list}_1$, then simply computes $\boldsymbol{\alpha}^{(0)} = \mathsf{coef}(\mathcal{S}^{(0)})$ and $\boldsymbol{\alpha}^{(1)} = \mathsf{coef}(\mathcal{S}^{(1)})$, queries $(1, \boldsymbol{\alpha}^{(0)}, \boldsymbol{\alpha}^{(1)})$ to its encryption oracle and returns the value it gets to $\mathcal{A}$.

If $i = 2$, $\mathcal{B}$ proceeds as follows: let $\mathsf{list}_b = (\mathcal{S}_1^{(b)}, \ldots, \mathcal{S}_q^{(b)})$ be the two lists stored by $\mathcal{B}$, for $b \in \{0, 1\}$ (each list contains respectively the $q$ left or right queries already made by $\mathcal{A}$ with index 1). $\mathcal{B}$ initializes an empty list $\mathsf{out}$ and applies the following process, termed $\mathsf{Pair}$, to sets $\mathcal{S}^{(0)}, \mathcal{S}^{(1)}$. It picks an element $s^{(0)}$ in $\mathcal{S}^{(0)}$ and checks for $i = 1, \ldots, q$ if $s^{(0)} \in \mathcal{S}_i^{(0)}$. Next, it searches for an element $s^{(1)} \in \mathcal{S}^{(1)}$ such that for $i = 1, \ldots, q$, $s^{(1)} \in \mathcal{S}_i^{(1)}$ if and only if $s^{(0)} \in \mathcal{S}_i^{(0)}$. Once such an element has been found, it computes $\boldsymbol{\beta}^{(0)} = \mathsf{exp}(s^{(0)})$ and $\boldsymbol{\beta}^{(1)} = \mathsf{exp}(s^{(1)})$, and queries $(2, \boldsymbol{\beta}^{(0)}, \boldsymbol{\beta}^{(1)})$ to its encryption oracle and adds the value it gets to the list $\mathsf{out}$. It then reiterates $\mathsf{Pair}$ to the sets $\mathcal{S}^{(0)} \setminus \{s^{(0)}\}, \mathcal{S}^{(1)} \setminus \{s^{(1)}\}$. Once every element has been handled, $\mathcal{B}$ shuffles $\mathsf{out}$ and sends a vector whose components are the elements of the list (in a random order) to $\mathcal{A}$. When $\mathcal{A}$ halts with some output, so does $\mathcal{B}$.

First, one needs to prove that the process $\mathsf{Pair}$ run by $\mathcal{B}$ to pair up elements from $\mathcal{S}^{(0)}$ with elements from $\mathcal{S}^{(1)}$ can always be done. As $\mathcal{A}$ is a polynomial-time adversary, it is clear that $\mathsf{Pair}$ is polynomial, as $q$ and $n$ are polynomial. Furthermore, by definition, $\mathcal{A}$ is restricted to only make sequences of queries such that at any time $\mathcal{L}_{\#^*}(\boldsymbol{\mathcal{S}}_1^{(0)}, \boldsymbol{\mathcal{S}}_2^{(0)}) = \mathcal{L}_{\#^*}(\boldsymbol{\mathcal{S}}_1^{(1)}, \boldsymbol{\mathcal{S}}_2^{(1)})$, where $\boldsymbol{\mathcal{S}}_i^{(b)}$ denote the series of left (if $b = 0$) or right (if $b = 1$) queries made at index $i$. This implies directly that $\mathsf{Pair}$ always terminates.

Second, one needs to prove that every query $(1, \boldsymbol{\alpha}^{(0)}, \boldsymbol{\alpha}^{(1)})$ or $(2, \boldsymbol{\beta}^{(0)}, \boldsymbol{\beta}^{(1)})$ made by $\mathcal{B}$ to its encryption oracle satisfies $\langle \boldsymbol{\alpha}^{(0)}, \boldsymbol{\beta}^{(0)} \rangle = 0$ if and only if $\langle \boldsymbol{\alpha}^{(1)}, \boldsymbol{\beta}^{(1)} \rangle = 0$. This is implied directly by the way process $\mathsf{Pair}$ is defined.

Finally, one needs to show that $\mathcal{B}$ simulates correctly the oracle, which is immediate from the description of Construction 2. This concludes the proof of Theorem 9. $\square$

**Corollary 11.** *Assuming* DLin*, there exists an $\mathcal{L}_{\#^*}$-indistinguishability secure 2-input functional encryption scheme for the function $f_{\#}$.*

*Proof.* Immediate from Corollary 8 and Theorem 9. $\square$

## 6 Order-Revealing Encryption with Limited Leakage

We finally describe how combining the previous tools, one obtain our main construction. As a preliminary, we explain how one can compare two numbers by simply checking the disjointness of two sets. Then, combining this technique with the construction from previous section, one proves the existence, under DLin, of an efficient order-revealing encryption scheme with limited leakage.

## 6.1 From Bitstrings to Sets

We define functions $\Sigma^0$ and $\Sigma^1$, taking as input an $n$-bit string $x$ and returning a set of prefixes, as follows:

$$\Sigma^b \colon x \in \{0,1\}^n \longmapsto \Sigma^b(x) = \big\{x_{n-1} \,\|\, \ldots \,\|\, x_{i+1} \,\|\, 1 \mid x_i = b\big\}_{0 \leq i \leq n-1} \ , \qquad (1)$$

for $b \in \{0,1\}$. That is, $\Sigma^1(x)$ returns the set of every prefix of $x$ that ends with a 1, and $\Sigma^0(x)$ returns the set of every $z \,\|\, 1$ such that $z \,\|\, 0$ is a prefix of $x$. It is easily seen that $\#\Sigma^1(x) = \mathsf{hw}(x)$ and that $\#\Sigma^0(x) = \mathsf{hw}(\bar{x})$. In particular, we have $\Sigma^0(1^n) = \Sigma^1(0^n) = \varnothing$ and thus $\#\Sigma^0(1^n) = \#\Sigma^1(0^n) = 0$. It is also immediate that $\#\Sigma^1(x \,\|\, \bar{x}) = \#\Sigma^0(x \,\|\, \bar{x}) = n$, for every $x \in \{0,1\}^n$.

Functions $\Sigma^0$ and $\Sigma^1$ are useful as they allow computing the relative order of two integers [17]. More precisely, we have:

**Lemma 12.** *Let $x, y$ be two integers such that $0 \leq x, y < 2^n$ and viewed as $n$-bit strings. Then*

$$x < y \iff \#\big(\Sigma^0(x) \cap \Sigma^1(y)\big) = 1 \quad \textit{and} \quad x \geq y \iff \#\big(\Sigma^0(x) \cap \Sigma^1(y)\big) = 0 \ .$$

The proof of the above lemma is detailed in Appendix D.

## 6.2 A Generic Transform from DRE to ORE

In this section is described a generic transform to obtain a dedicated 2-input functional encryption scheme for the function $f_< \colon (x,y) \mapsto \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$ from any dedicated 2-input functional encryption scheme for the function $f_\#$. This transform simply relies on the above technique. As we want the size of the sets encrypted to be some fixed constant, instead of directly encrypting the sets $\Sigma^0(x)$ or $\Sigma^1(x)$, one encrypts the sets $\Sigma^0(x \,\|\, \bar{x})$ or $\Sigma^1(x \,\|\, \bar{x})$, which are both of size $n$ if $x$ is an $n$-bit integer. It is very easy to see that Lemma 12 still holds even if we replace $\Sigma^0(x)$ and $\Sigma^1(y)$ by $\Sigma^0(x \,\|\, \bar{x})$ and $\Sigma^1(y \,\|\, \bar{y})$ respectively.

Let $\mathcal{D2IFE}_\# = (\mathsf{Setup}_\#, \mathsf{Enc}_\#, \mathsf{Eval}_{f_\#})$ be a dedicated 2-input functional encryption scheme for the function $f_\#$.

**Construction 3.** We build a dedicated 2-input functional encryption scheme $\mathcal{D2IFE}_< = (\mathsf{Setup}_<, \mathsf{Enc}_<, \mathsf{Eval}_{f_<})$ for the function $f_<$ as follows:

- $\mathsf{Setup}_<$ takes as input the security parameter $\kappa$ and outputs $\mathsf{Setup}_\#(1^\kappa) = \mathsf{sk}$;
- $\mathsf{Enc}_<$ takes as input an index $i \in \{1,2\}$, a secret key $\mathsf{sk}$, and a message $x$ and outputs:

$$\mathsf{Enc}_<(i, \mathsf{sk}, x) = \begin{cases} \mathsf{Enc}_\#(1, \mathsf{sk}, \Sigma^0(x \,\|\, \bar{x})) & \text{if } i = 1 \\ \mathsf{Enc}_\#(2, \mathsf{sk}, \Sigma^1(x \,\|\, \bar{x})) & \text{if } i = 2 \end{cases} ;$$

- $\mathsf{Eval}_{f_<}$ takes as input a pair of ciphertexts $(\mathsf{ct}_1, \mathsf{ct}_2)$ encrypted with index 1 and 2 respectively, and returns $\mathsf{Eval}_{f_\#}(\mathsf{ct}_1, \mathsf{ct}_2)$.

Correctness. The correctness easily follows from the correctness of $\mathcal{D2IFE}_{\#}$ and from Lemma 12.

Security. Security immediately follows from the security of $\mathcal{D2IFE}_{\#}$ and the leakage is simply the leakage associated of $\mathcal{D2IFE}_{\#}$ applied to the encrypted sets, which are either $\Sigma^0(x \,\|\, \bar{x})$ or $\Sigma^1(x \,\|\, \bar{x})$.

Let $\mathcal{L}$ denote a leakage such that $\mathcal{D2IFE}_{\#}$ is $\mathcal{L}$-indistinguishability secure. Then, we define the leakage of Construction 3 as:

$$\mathscr{L}_{\mathcal{L}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathcal{L}(\Sigma^0(\boldsymbol{x}_1), \Sigma^1(\boldsymbol{x}_2)) \ ,$$

where $\boldsymbol{x}_i = (x_{i,1}, \dots, x_{i,q_i})$ is the sequence of integers encrypted with index $i$, for $i \in \{1,2\}$, and where $\Sigma^0(\boldsymbol{x}_1) = (\Sigma^0(x_{1,1} \,\|\, \bar{x}_{1,1}), \dots, \Sigma^0(x_{1,q_1} \,\|\, \bar{x}_{1,q_1}))$, and $\Sigma^1(\boldsymbol{x}_2) = (\Sigma^1(x_{2,1} \,\|\, \bar{x}_{2,1}), \dots, \Sigma^1(x_{2,q_2} \,\|\, \bar{x}_{2,q_2}))$.

**Theorem 13.** *Assuming there exists an $\mathcal{L}$-indistinguishability secure dedicated 2-input functional encryption scheme for the function $f_{\#}$, then there exists an $\mathscr{L}_{\mathcal{L}}$-indistinguishability secure 2-input functional encryption scheme for the function $f_{<}$.*

*Proof.* Let $\mathcal{A}$ be an adversary against the $\mathscr{L}_{\mathcal{L}}$-indistinguishability security of $\mathcal{D2IFE}_{<}$ obtained via Construction 3. Then, one can build an adversary $\mathcal{B}$ against the $\mathcal{L}$-indistinguishability security of $\mathcal{D2IFE}_{\#}$ as follows: when $\mathcal{A}$ makes a query $(i, x^{(0)}, x^{(1)})$ to the encryption oracle, $\mathcal{B}$ does the following. It computes $(\mathcal{S}^{(0)}, \mathcal{S}^{(1)}) = (\Sigma^0(x^{(0)}), \Sigma^0(x^{(1)}))$ if $i = 1$ or $(\mathcal{S}^{(0)}, \mathcal{S}^{(1)}) = (\Sigma^1(x^{(0)}), \Sigma^1(x^{(1)}))$ if $i = 2$, queries $(i, \mathcal{S}^{(0)}, \mathcal{S}^{(1)})$ to its encryption oracle, and returns the value it gets to adversary $\mathcal{A}$. When adversary $\mathcal{A}$ halts with output $b$, so does $\mathcal{B}$. It is clear that the simulation is perfect. The only thing that one needs to prove is that the sequence of queries made by $\mathcal{B}$ is possible. This is immediate from the definition of $\mathscr{L}_{\mathcal{L}}$. Theorem 13 follows. $\square$

*Remark 14.* One could also prove in a very similar manner that the obtained construction is simulation-secure assuming the underlying scheme $\mathcal{D2IFE}_{\#}$ is simulation-secure.

**Corollary 15.** *Assuming* DLin*, there exists a $\mathscr{L}_{\mathcal{L}_{\#^*}}$-indistinguishability secure 2-input functional encryption scheme for the function $f_{<}$.*

*Proof.* Immediate from Corollary 11 and Corollary 13. $\square$

Detail of the Leakage. For the sake of clarity, here is a more intelligible description of the leakage of our resulting order-revealing encryption scheme. For simplicity, let us consider a sequence of integers $(x_1, \dots, x_q)$ encrypted respectively with index 1 and an integer $y$ encrypted with index 2, our encryption scheme reveals:

1. the order of $x_i$ relatively to $y$, for any $1 \le i \le q$;

2. for every subset $\mathcal{S}$ of $\{x_1, \ldots, x_q\}$ of size at least 2, whether there exists a bitstring $z$ such that $z \,\|\, 0$ is a common prefix of every $x \in \mathcal{S}$ and $z \,\|\, 1$ is a prefix of $y$.

The first part is exactly what we want to reveal, but the second part reveals extra information. However, as shown in Lemma 12, for every $x, y$, there exists at most one such bitstring $z$ such that $z \,\|\, 0$ is a prefix of $x$ and $z \,\|\, 1$ is a prefix of $y$ (and in that case, $x < y$). Thus, there is no such bistring $z$ for most subsets $\mathcal{S} \subseteq \{x_1, \ldots, x_q\}$, and the second part does not reveal much information.

The general case where multiple integers are encrypted with index 2 simply reveals the two information detailed above for every $y$ encrypted with index 2.

Possible Security Trade-Off. Since the leakage only depends on the number of common prefixes, a simple way to restrict this leakage is to first encrypt the messages with an order-preserving encryption scheme.

Another way to reduce the leakage is to improve the underlying construction of 2-input dedicated functional encryption for cardinality of intersection. We recall that a construction for cardinality of intersection with ideal leakage would immediately imply a solution for comparison with ideal leakage. In particular, one could use multivariate polynomials instead of univariate polynomials to check the intersection.

Intuitively, instead of testing $P_{\mathcal{A}}(b) = 0$ with $P_{\mathcal{A}}(X) = \prod_{a \in \mathcal{A}}(X - a)$ for checking if $b \in \mathcal{A}$, one could test $P^{(k)}_{\mathcal{A}}(b_1, \ldots, b_k) = 0$ with $P^{(k)}_{\mathcal{A}}(X_1, \ldots, X_k) = \prod_{i \in [k]} \prod_{a \in \mathcal{A}}(X_i - a)$. The leakage now only reveals if $k$-tuple of elements in $\mathcal{B}$ have an non-empty intersection with $\mathcal{A}$, but the size of the ciphertexts blow-up, since one needs to compute inner-products of vectors of length $(n+1)^k$ instead of vectors of length $n + 1$.

## 7 Concluding Remarks

### 7.1 Applications

Membership Testing on a Database and Searchable Encryption. Our notion of dedicated 2-input functional encryption for the function $f_{\#}$ naturally yields a solution to test whether some private data is already in a database stored by a given server. Indeed, one could split the database into distinct sets $\mathcal{S}_1, \ldots, \mathcal{S}_q$ of fixed size $n$ and storing encryptions $\mathsf{Enc}_{\#}(1, \mathsf{sk}, \mathsf{coef}(\mathcal{S}_i))$ for $i \in [q]$. Then, one can simply send to the server $\mathsf{Enc}_{\#}(2, \mathsf{sk}, \mathsf{exp}(a))$ so it can learn whether $a$ is already in the database. One could also use this method with a plaintext $x$ being a tag used to ask the server to return every encrypted data with the same tag.

Range Queries. Our notion of dedicated 2-input functional encryption for the function $f_{<}$ allows one to perform efficient range queries on a database. One could indeed store encryptions $\mathsf{Enc}_{<}(1, \mathsf{sk}, x)$ on the server, and makes queries of the form $\mathsf{Enc}_{<}(2, \mathsf{sk}, a), \mathsf{Enc}_{<}(2, \mathsf{sk}, b)$ to get encrypted data $x \in [a; b)$. In particular, as our notion is "asymmetric", the server learns only a few extra information, while classical order-revealing encryption let the server knows the complete order of the elements.

### 7.2 Conclusion

In this paper, we studied particular cases of multi-input functional encryption. To obtain pratical constructions, we allow our constructions to leak a bit more information than just the offered functionality. This results in efficient constructions for computing inner-product, cardinality of intersection, and comparison, under standard assumptions.

Our more general approach enables, in particular, to build an efficient order-revealing encryption scheme with very limited leakage compared to the information leaked by the recent construction proposed in [11]. Of independent interest, we also propose a construction with ideal security for polynomial-size message space.

## References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Paris, France, June 13–18, 2004. ACM Press.
2. S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, and A. Sahai. Functional encryption and property preserving encryption: New definitions and positive results. Cryptology ePrint Archive, Report 2013/744, 2013. `http://eprint.iacr.org/2013/744`.
3. S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, and A. Sahai. On the practical security of inner product functional encryption. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 777–798. Springer, Heidelberg, Mar. / Apr. 2015.
4. A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, Heidelberg, Apr. 2009.
5. A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 578–595. Springer, Heidelberg, Aug. 2011.
6. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, Aug. 2004.
7. D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 563–594. Springer, Heidelberg, Apr. 2015.
8. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, Mar. 2011.
9. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, Feb. 2007.
10. S. Chatterjee and M. P. L. Das. Property preserving symmetric encryption revisited. LNCS, pages 658–682. Springer, Heidelberg, Dec. 2015.

11. N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In *FSE 2016*, LNCS, Bochum, Germany, Mar. 20–23, 2016. Springer, Heidelberg. Full version available as Cryptology ePrint Archive, Report 2015/1125, `http://eprint.iacr.org/2015/1125`.
12. J. H. Cheon, P.-A. Fouque, C. Lee, B. Minaud, and H. Ryu. Cryptanalysis of the new CLT multilinear map over the integers. Cryptology ePrint Archive, Report 2016/135, 2016. `http://eprint.iacr.org/2016/135`.
13. J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the multilinear map over the integers. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, Apr. 2015.
14. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 79–88. ACM Press, Oct. / Nov. 2006.
15. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.
16. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, Apr. 2008.
17. H.-Y. Lin and W.-G. Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 456–466. Springer, Heidelberg, June 2005.
18. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, Aug. 2010.
19. O. Pandey and Y. Rouselakis. Property preserving symmetric encryption. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 375–391. Springer, Heidelberg, Apr. 2012.
20. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.
21. E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473. Springer, Heidelberg, Mar. 2009.

## A Proofs of Lemmata in Section 2.3

### A.1 Proof of Lemma 4

Let $\mathcal{DMIFE}$ be an $\mathcal{L}$-simulation secure property-revealing encryption scheme. Then, there exists a simulator $\mathcal{S}$ such that for any adversary $\mathcal{A}$, the distributions $\mathsf{Real}_{\mathcal{A}}^{\mathcal{DMIFE}}$ and $\mathsf{Sim}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\mathcal{DMIFE}}$ are computationally indistinguishable.

Let $\mathcal{B}$ denote an adversary against the $\mathcal{L}$-indistinguishability security of $\mathcal{DMIFE}$ that makes a sequence of queries $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$. Then, the sequence of ciphertexts $(\mathbf{ct}_1, \ldots, \mathbf{ct}_k)$ is computationally instinguishable from the distribution output by the simulator $\mathcal{S}$ which is computed only from $\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$. Hence, as the latter distribution does not depend on the messages but only on their leakage, for any two sequences of queries $(\boldsymbol{x}_1^{(0)}, \ldots, \boldsymbol{x}_k^{(1)})$ and $(\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_k^{(1)})$ such that $\mathcal{L}(\boldsymbol{x}_1^{(0)}, \ldots, \boldsymbol{x}_k^{(0)}) = \mathcal{L}(\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_k^{(1)})$, the distributions of ciphertexts are computationally indistinguishable. Lemma 4 follows. $\square$

### A.2 Proof of Lemma 5

Indistinguishability Security. Let $\mathcal{DMIFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$ be an $\mathcal{L}$-indistinguishability secure "asymmetric" property-revealing encryption scheme for a $k$-ary property $P$. Then, $\mathcal{DMIFE}' = (\mathsf{Setup}, \mathsf{Enc}', \mathsf{Eval}_f)$ with $\mathsf{Setup}, \mathsf{Eval}_f$ being the same as in $\mathcal{DMIFE}$ and $\mathsf{Enc}'(\mathsf{sk}, x) = (\mathsf{Enc}(1, \mathsf{sk}_1, x), \ldots, \mathsf{Enc}(k, \mathsf{sk}_k, x))$, is a $\mathsf{sym}_{\mathcal{L}}$-indistinguishability secure "symmetric" property-revealing encryption scheme for the property $P$. The proof is immediate: given an adversary $\mathcal{A}$ against the indistinguishability security of $\mathcal{DMIFE}'$ that makes a sequence of queries $((x_1^{(0)}, x_1^{(1)}), \ldots, (x_q^{(0)}, x_q^{(1)}))$, one can simply build an adversary $\mathcal{B}$ against the indistinguishability security of $\mathcal{DMIFE}$ as follows: when $\mathcal{A}$ makes a query $(x^{(0)}, x^{(1)})$, $\mathcal{B}$ makes the query $(i, x^{(0)}, x^{(1)})$ to its oracle for all $i = 1, \ldots, k$ and returns the tuple of $k$ ciphertexts obtained to $\mathcal{A}$. When $\mathcal{A}$ halts with output $b$, so does $\mathcal{B}$. The only thing that one needs to prove is that the sequence of queries made by $\mathcal{B}$ is possible. This is immediate from the fact that $\mathcal{A}$ is restricted to make sequences of queries $((x_1^{(0)}, x_1^{(1)}), \ldots, (x_q^{(0)}, x_q^{(1)}))$ such that $\mathsf{sym}_{\mathcal{L}}(\boldsymbol{x}^{(0)}) = \mathsf{sym}_{\mathcal{L}}(\boldsymbol{x}^{(1)})$.

Simulation Security. Let $\mathcal{DMIFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$ be an $\mathcal{L}$-simulation secure "asymmetric" property-revealing encryption scheme for a $k$-ary property $P$. Then, $\mathcal{DMIFE}' = (\mathsf{Setup}, \mathsf{Enc}', \mathsf{Eval}_f)$ with $\mathsf{Setup}, \mathsf{Eval}_f$ being the same as in $\mathcal{DMIFE}$ and $\mathsf{Enc}'(\mathsf{sk}, x) = (\mathsf{Enc}(1, \mathsf{sk}_1, x), \ldots, \mathsf{Enc}(k, \mathsf{sk}_k, x))$ is a $\mathsf{sym}_{\mathcal{L}}$-simulation secure symmetric property-revealing encryption scheme for the property $P$. The proof is immediate: given an adversary $\mathcal{A}$ against the $\mathsf{sym}_{\mathcal{L}}$-simulation security of $\mathcal{DMIFE}'$ that makes a sequence of queries $(x_1, \ldots, x_q)$, one can simply build an adversary $\mathcal{B}$ against the simulation security of $\mathcal{DMIFE}$ as follows: when $\mathcal{A}$ makes a query $x$, $\mathcal{B}$ makes the query $(i, x)$ to its oracle for all $i = 1, \ldots, k$. The $\mathcal{L}$-simulation security of $\mathcal{DMIFE}$ guarantees that the distribution of ciphertexts obtained is indistinguishable from the one computed by the simulator $\mathcal{S}$ given only $\mathcal{L}(\boldsymbol{x}, \ldots, \boldsymbol{x})$.

Lemma 5 easily follows. $\square$

## B Proof of Theorem 6

Let $\mathcal{A} = (\mathcal{A}_0, \ldots, \mathcal{A}_{q_1+q_2})$ be an adversary against the $\mathcal{L}_{<,=}$-simulation security of $\mathcal{D2IFE}_<$, described in Construction 1, where $q_1, q_2$ are polynomial in the security parameter and correspond to the number of queries made with index 1 and 2 respectively. To prove security, one needs to build a simulator $\mathcal{S} = (\mathcal{S}_0, \ldots, \mathcal{S}_{q_1+q_2})$

such that the distributions output by the experiments $\mathsf{Real}_{\mathcal{A}}^{\mathcal{D2IFE}_<}$ and $\mathsf{Sim}_{\mathcal{A},\mathcal{S},\mathcal{L}_{<,=}}^{\mathcal{D2IFE}_<}$ are computationally indistinguishable.

The distribution output by experiment $\mathsf{Real}_{\mathcal{A}}^{\mathcal{D2IFE}_<}$ consists in a sequence of ciphertexts $(\mathbf{ct}_1, \mathbf{ct}_2)$ which are $q_1$ ciphertexts encrypted with index 1 and $q_2$ ciphertexts encrypted with index 2. Let us recall that:

$$\mathsf{Enc}_<(i, K, x) = \begin{cases} \mathsf{shuffle}(F_K(x, x+1), \ldots, F_K(x, x+N-1)) & \text{if } i = 1 \\ \mathsf{shuffle}(F_K(0, x), \ldots, F_K(N-1, x)) & \text{if } i = 2 \end{cases}.$$

Then, under the PRF security of $F$, the distribution output by $\mathsf{Real}_{\mathcal{A}}^{\mathcal{D2IFE}_<}$ is computationally indistinguishable from the distribution $\mathsf{Hyb}$ where the ciphertexts are computed using a trully random function $f\colon \mathcal{D} \to \mathcal{R}$.

We now describe our simulator $\mathcal{S}$. $\mathcal{S}_0$ initializes an empty table $\mathsf{T}$ of size $(q_1 + q_2) \times (N + 2)$ and outputs $\mathsf{st}_{\mathcal{S}} = \mathsf{T}$. At any time, the $i$-the row of $\mathsf{T}$ has the following form: the first column contains whether the $i$-th query of $\mathcal{A}$ is a query with index 1 or 2. The second column will be a counter starting from 3. The remaining columns contain the $N$ components of the encryption returned to the adversary.

When $\mathcal{A}_t$ asks for an encryption of a message $x_t$, $\mathcal{S}_t$ does the following: let us denote by $i$ and $j$ the numbers of queries made by the adversary to oracle with index 1 and 2 respectively before step $t$, so $t = i + j + 1$, and let $(x_{1,1}, \ldots, x_{1,i})$ and $(x_{2,1}, \ldots, x_{2,j})$ denote these two series of queries.

We assume the adversary does not make twice the same query to the same encryption algorithm. If it does, one can simply adapt the simulator as follows: assume that there exists $k < t$ such that $\mathcal{A}_k$ made the same query $x_t$ as $\mathcal{A}_t$ to the same encryption algorithm. Then, $\mathcal{S}_t$ returns $\mathsf{shuffle}(\mathsf{T}[k, 3], \ldots, \mathsf{T}[k, N+2])$.

Let us now assume that all queries made by $\mathcal{A}$ to encryption oracle are different.

Let us assume that the adversary asks makes a query $(1, x_t)$ and let $\boldsymbol{x}_1 = \boldsymbol{x}_1.\mathsf{append}(x_t)$. $\mathcal{S}_t$ is executed on input $\mathsf{st}_{\mathcal{S}}$ and $\mathcal{L}_{<,=}(\boldsymbol{x}_1, \boldsymbol{x}_2)$. In particular, $\mathcal{S}_t$ knows, for every $1 \le k \le j$ whether $x_t < x_{2,k}$. $\mathcal{S}_t$ let $\mathsf{T}[t, 1] \leftarrow 1$. Let $n_1, \ldots, n_j$ denote the $j$ indices of the rows of $\mathsf{T}$ that corresponds to queries with index 2. Let $c = 3$. Then, for any $1 \le k \le j$, $\mathcal{S}_t$ does the following: if $x_t < x_{2,k}$, $\mathcal{S}_t$ lets $\mathsf{T}[t, c] \leftarrow \mathsf{T}[n_k, \mathsf{T}[n_k, 2]]$, $\mathsf{T}[n_k, 2] \leftarrow \mathsf{T}[n_k, 2] + 1$, $c \leftarrow c + 1$; if $x_t \ge x_{2,k}$, $\mathcal{S}_t$ does nothing. Finally, $\mathcal{S}_t$ sets $\mathsf{T}[t, 2] \leftarrow c$ picks $N + 3 - c$ values at random in $\mathcal{R}$ and completes the last $N + 3 - c$ empty cases of the $t$-th row of $\mathsf{T}$ with these values. It finally outputs $\mathsf{shuffle}(\mathsf{T}[t, 3], \ldots, \mathsf{T}[t, N+2])$ as the ciphertext $\mathsf{ct}_{1,i+1}$.

Let us now assume that the adversary makes a query $(2, x_t)$ and let $\boldsymbol{x}_2 = \boldsymbol{x}_2.\mathsf{append}(x_t)$. $\mathcal{S}_t$ is executed on input $\mathsf{st}_{\mathcal{S}}$ and $\mathcal{L}_{<,=}(\boldsymbol{x}_1, \boldsymbol{x}_2)$. In particular, $\mathcal{S}_t$ knows, for every $1 \le k \le i$ whether $x_{1,k} < x_t$. $\mathcal{S}_t$ let $\mathsf{T}[t, 1] \leftarrow 2$. Let $n_1, \ldots, n_i$ denote the $i$ indices of the rows of $\mathsf{T}$ that corresponds to queries with index 1. Let $c = 3$. Then, for any $1 \le k \le i$, $\mathcal{S}_t$ does the following: if $x_{1,k} < x_t$, $\mathcal{S}_t$ lets $\mathsf{T}[t, c] \leftarrow \mathsf{T}[n_k, \mathsf{T}[n_k, 2]]$, $\mathsf{T}[n_k, 2] \leftarrow \mathsf{T}[n_k, 2] + 1$, $c \leftarrow c + 1$; if $x_{1,k} \ge x_t$, $\mathcal{S}_t$ does nothing. Finally, $\mathcal{S}_t$ sets $\mathsf{T}[t, 2] \leftarrow c$ picks $N + 3 - c$ values at random in $\mathcal{R}$ and completes the last $N + 3 - c$ empty cases of the $t$-th row of $\mathsf{T}$ with these values. It finally outputs $\mathsf{shuffle}(\mathsf{T}[t, 3], \ldots, \mathsf{T}[t, N+2])$ as the ciphertext $\mathsf{ct}_{2,j+1}$.

Then, it is immediate that the distribution described above is identical to the distribution of $\mathsf{Hyb}$, which is computationally indistinguishable from the distribution of $\mathsf{Real}_{\mathcal{A}}^{\mathcal{D2IFE}_<}$. Theorem 6 follows. $\square$

## C Additional Definitions

### C.1 DLin assumption

We recall the definition of the DLin problem in a group $\mathbb{G} = \langle g \rangle$ of order $N$, which states the hardness of distinguishing whether $z = g^{w_1+w_2}$ from a random group element, when given a tuple $(g, g^{a_1}, g^{a_2}, g^{a_1 w_1}, g^{a_2 w_2}, z)$, where $a_i, w_i \overset{\$}{\leftarrow} \mathbb{Z}_N$ for $i = 1, 2$. The DLin assumption corresponds to the hardness of the DLin problem.

### C.2 Predicate Encryption and Inner Product Encryption

**Definition 16 (Secret-Key Predicate Encryption).** *A* secret-key predicate encryption *scheme is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{TokenGen}, \mathsf{Enc}, \mathsf{Dec})$*, defined as follows:*

– $\mathsf{Setup}$ *takes as input the security parameter* $1^\kappa$ *and ouputs a secret key* $\mathsf{sk}$*;*
– $\mathsf{TokenGen}$ *takes as inputs a secret key* $\mathsf{sk}$ *and a predicate* $\boldsymbol{P}$ *and outputs a token* $\mathsf{tk}_P$*;*
– $\mathsf{Enc}$ *takes as inputs a secret key* $\mathsf{sk}$ *and an attribute* $I$ *and a message* $x$ *and outputs a ciphertext* $\mathsf{ct}_{I,x}$*;*
– $\mathsf{Dec}$ *takes as input a token* $\mathsf{tk}_P$ *and a ciphertext* $\mathsf{ct}_{I,x}$ *and outputs* $x$ *or* $\perp$*.*

*For correctness, we require that for any* $\mathsf{sk} \overset{\$}{\leftarrow} \mathsf{Setup}(1^\kappa)$ *and any pair* $(\mathsf{tk}_P, \mathsf{ct}_{I,x})$ *with* $\mathsf{ct}_{I,x} = \mathsf{Enc}(\mathsf{sk}, I, x)$ *and* $\mathsf{tk}_P = \mathsf{TokenGen}(\mathsf{sk}, P)$*, then* $\mathsf{Dec}(\mathsf{tk}_P, \mathsf{ct}_{I,x}) = x \iff P(I) = 1$*.*

Security. A secret-key predicate encryption scheme is fully-secure if a token $\mathsf{tk}$ (resp. a ciphertext $\mathsf{ct}$) reveals nothing about the predicate (resp. attribute, message) vector beyond the value of the predicate on queried attributes (resp. the values of queried predicates on the attribute). This security notion is defined as follows: the adversary has access to two left-or-right oracles that can be adaptively queried with pair of predicates $(P_0, P_1)$ (resp. pair of attributes and messages $((I_0, x_0), (I_1, x_1))$) to get token $\mathsf{tk}_{P_b}$ (resp. ciphertext $\mathsf{ct}_{I_b, x_b}$), where $b$ is a fixed bit chosen at random in the **Initialize** procedure. At the end, the adversary outputs a bit $b'$ and wins if $b = b'$. Once again, the adversary is restricted to avoid trivial attacks. Hence, for any sequence of messages $(((I_0^{(1)}, x_0^{(1)}), (I_1^{(1)}, x_1^{(1)})), \ldots, ((I_0^{(q_1)}, x_0^{(q_1)}), (I_1^{(q_1)}, x_1^{(q_1)})))$ and predicates $(P_0^{(1)}, P_1^{(1)}), \ldots, (P_0^{(q_2)}, P_1^{(q_2)})$, we require that for all $1 \leq i \leq q_1$ and $1 \leq j \leq q_2$, $P_0^{(j)}(I_0^{(i)}) = 1 \iff P_1^{(j)}(I_1^{(i)})$ and if so, $x_0^{(i)} = x_1^{(i)}$.

**Definition 17 (Secret-Key Inner Product Encryption).** *A* secret-key inner product encryption *scheme is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{TokenGen}, \mathsf{Enc}, \mathsf{Query})$*, defined as follows:*

– $\mathsf{Setup}$ *takes as input the security parameter* $1^\kappa$ *and ouputs a secret key* $\mathsf{sk}$*;*
– $\mathsf{TokenGen}$ *takes as inputs a secret key* $\mathsf{sk}$ *and a predicate vector* $\boldsymbol{y}$ *and outputs a token* $\mathsf{tk}_{\boldsymbol{y}}$*;*
– $\mathsf{Enc}$ *takes as inputs a secret key* $\mathsf{sk}$ *and an attribute (or plaintext) vector* $\boldsymbol{x}$ *and outputs a ciphertext* $\mathsf{ct}_{\boldsymbol{x}}$*;*
– $\mathsf{Query}$ *takes as input a token* $\mathsf{tk}_{\boldsymbol{y}}$ *and a ciphertext* $\mathsf{ct}_{\boldsymbol{x}}$ *and outputs* $0$ *or* $1$*.*

*For correctness, we require that for any* $\mathsf{sk} \overset{\$}{\leftarrow} \mathsf{Setup}(1^\kappa)$ *and any pair* $(\mathsf{tk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}})$ *with* $\mathsf{ct}_{\boldsymbol{x}} = \mathsf{Enc}(\mathsf{sk}, \boldsymbol{x})$ *and* $\mathsf{tk}_{\boldsymbol{y}} = \mathsf{TokenGen}(\mathsf{sk}, \boldsymbol{y})$*, then* $\mathsf{Query}(\mathsf{tk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}}) = 1 \iff \langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$*.*

Security. A secret-key inner product encryption scheme is fully-secure if a token $\mathsf{tk}$ (resp. a ciphertext $\mathsf{ct}$) reveals nothing about the predicate (resp. attribute) vector beyond the value of the predicate on queried attributes (resp. the values of queried predicates on the attribute). This security notion is defined as follows: the adversary has access to two left-or-right oracles that can be adaptively queried with pair of predicates $(\boldsymbol{y}_0, \boldsymbol{y}_1)$ (resp. pair of attributes $(\boldsymbol{x}_0, \boldsymbol{x}_1)$) to get token $\mathsf{tk}_{\boldsymbol{y}_b}$ (resp. ciphertext $\mathsf{ct}_{\boldsymbol{x}_b}$), where $b$ is a fixed bit chosen at random in the **Initialize** procedure. At the end, the adversary outputs a bit $b'$ and wins if $b = b'$. Once again, the adversary is restricted to avoid trivial attacks. Hence, for any sequence of queries $(\boldsymbol{x}_0^{(1)}, \boldsymbol{x}_1^{(1)}), \ldots, (\boldsymbol{x}_0^{(q_1)}, \boldsymbol{x}_1^{(q_1)}), (\boldsymbol{y}_0^{(1)}, \boldsymbol{y}_1^{(1)}), \ldots, (\boldsymbol{y}_0^{(q_2)}, \boldsymbol{y}_1^{(q_2)})$, we require that for all $1 \leq i \leq q_1$ and $1 \leq j \leq q_2$, $\langle \boldsymbol{x}_0^{(i)}, \boldsymbol{y}_0^{(j)} \rangle = 0 \iff \langle \boldsymbol{x}_1^{(i)}, \boldsymbol{y}_1^{(j)} \rangle = 0$.

## D Proof of Lemma 12

Suppose first that $x \leq y$. Then there must exist a biggest index $i \in \{0, \ldots, n-1\}$ such that $x_i = 0$ and $y_i = 1$ and, if $i \neq n-1$, $x_{n-1} \parallel \ldots \parallel x_{i+1} = y_{n-1} \parallel \ldots \parallel y_{i+1}$. If $i = n-1$ this implies $\{1\} \subseteq \Sigma^0(x) \cap \Sigma^1(y)$, and if $i \neq n-1$ this implies $\{x_{n-1} \parallel \ldots \parallel x_{i+1} \parallel 1\} \subseteq \Sigma^0(x) \cap \Sigma^1(y)$.

Suppose now that $\Sigma^0(x) \cap \Sigma^1(y) \neq \varnothing$. If $\Sigma^0(x) \cap \Sigma^1(y) = \{1\}$ then it is cleat that $x < y$. Otherwise, there exists an index $i \in \{0, \ldots, n-2\}$ such that $x_{n-1} \parallel \ldots \parallel x_{i+1} \parallel 1 \in \Sigma^0(x) \cap \Sigma^1(y)$. This means that $x_i = 0$, $y_i = 1$, and $x_{n-1} \parallel \ldots \parallel x_{i+1} = y_{n-1} \parallel \ldots \parallel y_{i+1}$, which in turn means $x < y$, since $\sum_{k=0}^{i-1} y_k \cdot 2^k \leq \sum_{k=0}^{i-1} 2^k = 2^i - 1 < 2^i$.

It remains to show that $\Sigma^0(x) \cap \Sigma^1(y)$ contains at most one element. Let us assume that $\#\big(\Sigma^0(x) \cap \Sigma^1(y)\big) > 1$. Then there exist $i, j \in \{0, \ldots, n-1\}$ distinct and such that we have $x_{n-1} \parallel \ldots \parallel x_{i+1} = y_{n-1} \parallel \ldots \parallel y_{i+1}$ as well as $x_{n-1} \parallel \ldots \parallel x_{j+1} = y_{n-1} \parallel \ldots \parallel y_{j+1}$, and $x_i = x_j = 0$, and $y_i = y_j = 1$. We can assume without loss of generality that $i > j$. Hence, $i \geq j+1$ and since $x_i = 0$ and $y_i = 1$, it is impossible that $x_{n-1} \parallel \ldots \parallel x_{j+1} = y_{n-1} \parallel \ldots \parallel y_{j+1}$. This concludes the proof. $\square$