Function-Revealing Encryption
Definitions and Constructions

Marc Joye[1] and Alain Passelègue[2]

[1] NXP Semiconductors (USA)
[2] UCLA (USA)

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, including comparing the relative order of two plaintexts from their encrypted form (order-revealing encryption).
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.
In this work, we investigate a simpler approach towards obtaining practical schemes for functions of particular interest. We introduce the notion of function-revealing encryption, a generalization of order-revealing encryption to any multi-input function as well as a relaxation of multi-input functional encryption. We then propose a simple construction of order-revealing encryption based on function-revealing encryption for simple functions, namely orthogonality testing and intersection cardinality. Our main result is an efficient order-revealing encryption scheme with limited leakage based on the standard DLin assumption.

Keywords: Order-revealing encryption; property-preserving encryption; multi-input functional encryption; function-revealing encryption.

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* [BSW11,SW05] is an extension of traditional encryption that enables an authorized user to compute a certain function of the plaintext. Each decryption key sk_f corresponds to a specific function f. Informally, this private key 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* [BW07,KSW08]. 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 introduced in [GGG+14,BLR+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) [BCO11,BLR+15]. 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 one to answer queries over encrypted data, including range queries, sorting queries, searching queries, and more [AKSX04,BCLO09].

From OPE to ORE. Order-revealing encryption evolved from *order-preserving encryption* (OPE) [BCLO09,BCO11], 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 [BCLO09] 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 [BLR+15], 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 [BLR+15] is hardly implementable since it relies on heavy cryptographic tools, namely $(\ell/2+1)$-way multilinear maps for comparing ℓ-bit values. Furthermore, and maybe more importantly, the underlying security assumption is questionable owing to the recent attacks mounted against multilinear maps [CFL+16,CHL+15].

ORE in Practice. A practical construction for order-revealing encryption is proposed in [CLWW16]. It merely requires a pseudorandom function F with output space $\{0,1,2\}$. The encryption under secret key K of an ℓ-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

$$c_i = \left[F\big(K, (i, m_1 m_2 \cdots m_{i-1} \| 0^{\ell-i})\big) + m_i\right] \bmod 3\,, \quad \text{for } 1 \le i \le \ell\ .$$

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, $1 \le i \le \ell$, such that $c'_i \neq c_i$. Then,

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

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 work, we investigate a new approach towards building efficient secret-key multi-input functional encryption. We propose the notion of function-revealing encryption, which can be viewed both as a generalization of the notion of property-preserving encryption [CD15,PR12] and as a specialization of the notion of multi-input functional encryption. Basically, a function-revealing 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 $f(x_1, \ldots, x_k)$. In particular, considering the comparison function defined as:

$$f_<\colon (x, y) \mapsto \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases},$$

our notion matches precisely the notion of order-revealing encryption.

We note that our general framework slightly generalizes the definition of order-revealing encryption, since the original definition is "symmetric" and ours is "asymmetric" (in the sense that our definition only allows to compare a ciphertext with index 1 with a ciphertext with index 2). This is without loss of generality since a symmetric scheme results immediately from an asymmetric scheme.

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 [BLR+15]), 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.

We then focus on the particular case of 2-ary functions (so the index is 1 or 2) and specifically on building efficient order-revealing encryption. Our main construction is an efficient order-revealing encryption scheme with limited leakage, under standard assumptions.

Our Techniques. We first show that one can build an order-revealing encryption scheme given only a function-revealing encryption scheme for the function computing the cardinality of the intersection of two sets $f_\#\colon (\mathcal{S}, \mathcal{T}) \mapsto \#(\mathcal{S} \cap \mathcal{T})$. This result follows from a fairly simple technique to compare two bitstrings. Consider two bitstrings of same length $x = x_1 \| \ldots \| x_n$ and $y = y_1 \| \ldots \| y_n$, then we have $x < y$ if and only if there exists $i \in [n]$ such that $x_j = y_j$ for every $j < i$ and $x_i = 0$ and $y_i = 1$. Thus, we have $x < y$ if and only if there exists a prefix $z \| 0$ of x with $z \in \{0,1\}^*$ such that $z \| 1$ is a prefix of y, and one can then compare x and y by checking if the sets $\{z \| 1 \mid z \| 0 \text{ is a prefix of } x\}$ and $\{z \| 1 \mid z \| 1 \text{ is a prefix of } y\}$ are disjoint.

The next step is then to construct a function-revealing encryption scheme for intersection cardinality. We show that one can build such a scheme with only limited leakage based on the existence of function-revealing encryption for the function checking the orthogonality of two vectors $f_\perp\colon (\vec{a}, \vec{b}) \mapsto \langle \vec{a}, \vec{b} \rangle = 0$ (this function outputs the value of the predicate $\langle \vec{a}, \vec{b} \rangle = 0$). This transformation relies on the following technique to compute the cardinality of the intersection. Consider two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$. A simple way to compute $\#(A \cap B)$ is to evaluate the polynomial $\prod_{i=1}^{n}(X - a_i)$ whose roots are the elements of A on every b_j for $j \in [m]$, and to return the number of times this evaluates to 0. This can be done by computing inner products, since $\prod_{i=1}^{n}(X - a_i)$ is a degree n polynomial that can be written as $\sum_{i=0}^{n} \alpha_i X^i$ and thus, we have

$$\prod_{i=1}^{n}(b - a_i) = \langle (\alpha_0, \ldots, \alpha_n), (1, b, b^2, \ldots, b^n) \rangle ,$$

so checking if this evaluates to 0 corresponds precisely to checking if the above vectors are orthogonal.

Finally, we show that one can build a function-revealing encryption scheme for $f_\perp$ under the standard DLin assumption. In particular, we show that any fully-secure predicate encryption scheme

for a class of predicate $\mathcal{P} = \{P_a : b \mapsto P(a, b)\}$ can be turned into a function-revealing encryption scheme for the function P.

Yet, there is a small catch in our transform from orthogonality to intersection cardinality. Indeed, the above function-revealing encryption scheme for $f_{\#}$ not only reveals the cardinality of the intersection, but also the elements of B that are in A, as each element b_i of B is encrypted separately (by encrypting the corresponding vector $(1, b_i, b_i^2, \ldots, b_i^n)$). In particular, even if b_i is hidden, intersecting A with B and A' with B might also reveal some information about the intersection of A and A' (for instance, if $b_i \in A$ and $b_i \in A'$, then we also learn that $A \cap A' \neq \varnothing$ which should not have been revealed). Thus, our construction reveals a bit more than what we would like ideally. We briefly discuss how one can reduce this leakage by reducing the efficiency of our construction (though the only way to obtain ideal leakage with our technique is by having exponential-size ciphertexts). Also, we note that our leakage *is* ideal if there is only one set A that is encrypted at index 1, whatever the number of sets $B, C, \ldots$ encrypted at index 2 (or more generally for bounded ciphertexts at index 1 and unbounded ciphertexts at index 2). Finally, since our transformation from intersection cardinality to relative order is generic, any improvement on the security of the underlying scheme for intersection cardinality (both in terms of efficiency and of leakage) would immediately results in an improved construction of order-revealing encryption.

Of independent interest, we also provide a very simple order-revealing encryption scheme achieving the best possible security for short messages, assuming only the existence of one-way functions.

Concurrent and Related Works. In a concurrent work by Lewi and Wu [LW16], the authors also proposed a similar construction for short messages. In this paper, we deviate from this approach and adopt a different one to obtain efficient constructions, while Lewi and Wu investigate how to extend the small domain construction to larger domains. They propose a transform from small domains to larger domains using random oracles. Intuitively, for plaintexts of length $n \cdot k$, they encrypt small blocks of k bits with the perfect scheme (whose complexity is exponential in k), and then compose with the scheme from [CLWW16] for each of the n blocks (the complexity of the latter scheme being linear in n). However, since the scheme from [CLWW16] has an important leakage, this transform also incurs a leakage. Specifically, the ciphertexts reveal the position of the first blocks of k bits on which corresponding plaintexts differ. Then, the bigger the blocks are, the less efficient the resulting scheme is, but also, the bigger the blocks are, the smaller the leakage is.

In another recent work by Durak, DuBuisson, and Cash [DDC16], the authors show that even ideal leakage can reveal important information for certain particular applications of ORE (e.g., when plaintexts come from particular distributions). This work emphasizes that an important leakage could be devastating, so reducing the leakage as much as possible (while preserving good efficiency due to the practical importance of ORE) is of prime interest. Our work proposes a first step towards obtaining smaller leakage (in particular achieving ideal leakage in restricted cases). A recent and concurrent work by Cash, Liu, O'Neill, and Zhang [CLOZ16] also makes a step in this direction. In this work, the authors construct an order-revealing encryption scheme with limited leakage under SXDH. Their construction is slightly more efficient than ours (basing our construction on current state-of-the-art fully-secure IPE [KT14]) but their leakage is slightly worse than ours, our construction beneficing from its asymmetry. They obtain a construction, based on pairings, that only leaks the equality pattern of the most significant differing bit (that is, for any 3 plaintexts m_0, m_1, m_2, whether the most significant differing bit of m_0 and m_1 is the same as the one of m_0 and m_2), while the construction from [CLWW16] reveals the position of the most significant differing bit. Despite the similarity of our results, both constructions are significantly different in terms of techniques, as [CLOZ16] is based on the work by Chenette *et al.* [CLWW16] while our work opens a

new path. In particular, any improvement of our building blocks (e.g., more efficient fully-secure IPE or construction for cardinality of intersection with smaller leakage) would immediately benefit to our ORE scheme.

Concerning multi-input functional encryption, a recent work by Brakerski, Komargodski, and Segev [BKS16], improved in [KS17], propose a more general approach that allows going from single-input functional encryption to t-input functional encryption in the private key-setting, as long as t is constant (or poly-logarithmic assuming quasi-polynomial security). In particular, this allows one to obtain function-revealing encryption scheme for functions with t-arity from LWE [GKP+13] (or from low-complexity PRG and public-key encryption [GVW12]) for a bounded number T of ciphertexts for one index and unbounded ciphertexts for the others (where the size of the ciphertexts grow with T and the depth of the circuit computing the function). A similar result for the case of 2-arity functions can also be obtained directly from the reusable garbled circuits construction from [GKP+13]. The general case with unbounded ciphertexts at both indexes remains out of reach since it requires unbounded-collusion functional encryption, which is not known from standard assumptions (and implies $i\mathcal{O}$ [BNPW16,KS17] up to subexponential security).

Finally, our notion of function-revealing encryption has also been defined in a recent concurrent work (as "revealing encryption") by Haagh, Ji, Li, Orlandi, and Song [HJL+17]. In this paper, the authors also propose a function-revealing scheme for the comparison of two vectors ($x_i < y_i$ or $x_i \geq y_i$ for all i or $\vec{x}$ and $\vec{y}$ are incomparable). Their construction is obtained by extending the order-revealing construction from [CLWW16] and thus implies as well an important leakage.

2 Definitions

2.1 Function-Revealing Encryption

We introduce the paradigm of function-revealing encryption (FRE), as a generalization of property-preserving encryption defined by Pandey and Rouselakis [PR12] as well as a weakening of the general notion of multi-input functional encryption [BLR+15,GGG+14]. Our notion assumes the private-key setting [SSW09] and corresponds to *dedicated* multi-input functional encryption schemes where the evaluation of the function is public (i.e., no functional secret key is involved).

Definition 1 (Function-Revealing Encryption). *A* function-revealing encryption scheme *for a k-ary function f consists of a tuple of algorithms $\mathcal{FRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$, defined below.*

- $\mathsf{Setup}(1^\kappa)$ *is a probabilistic algorithm that takes as input the security parameter 1^κ and outputs a secret key sk (and public parameters pp—including the message space $\mathcal{M}$).*
- $\mathsf{Enc}(i, \mathsf{sk}, x)$ *takes as input an index $i \in [k]$, a key sk, and a message $x \in \mathcal{M}$. It outputs a ciphertext ct. Index i indicates that the output ciphertext 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 Eval_f procedure. We could define a "symmetric" version of function-revealing encryption where the encryption algorithm Enc no longer takes in an index $i \in [k]$ so that a ciphertext can be used in any input position for the 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. As specified in the introduction, we focus on the three following functions:
   - $f_{\perp}\colon (\vec{a},\vec{b}) \mapsto \begin{cases} 1 & \text{if } \langle \vec{a},\vec{b}\rangle = 0 \\ 0 & \text{otherwise} \end{cases}$;
   - $f_{\#}\colon (\mathcal{S},\mathcal{T}) \mapsto \#(\mathcal{S}\cap\mathcal{T})$;
   - $f_{<}\colon (x,y) \mapsto \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$.

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 [AAB+15,CD15,CGKO06,PR12].

The two notions are defined relatively to a leakage function $\mathcal{L}$. As a FRE scheme for a function f has to reveal, via the 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 $\vec{x}_1,\ldots,\vec{x}_k$ of $q_1,\ldots,q_k$ messages respectively —one vector of messages per position in the input of the function, so $\vec{x}_i = (x_{i,1},\ldots,x_{i,q_i})$— is defined as:*

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

$\mathcal{L}$-Indistinguishability Security. A FRE 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 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, **Finalize**$(b') = 1$. In order to prevent trivial attacks (i.e., attacks resulting from the leakage function), the adversary is restricted as follows. If $((x^{(0)}_{i,1},x^{(1)}_{i,1}),\ldots,(x^{(0)}_{i,q_i},x^{(1)}_{i,q_i}))$ denotes the sequence of q_i queries made with index i to the **LoR** oracle then, letting $\vec{x}^{(t)}_i = (x^{(t)}_{i,1},\ldots,x^{(t)}_{i,q_i})$ for $t \in \{0,1\}$, the sequence of queries made by the adversary has to satisfy:

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

$\mathcal{L}$-Simulation Security. A FRE 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 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 $\mathsf{Real}^{\mathcal{FRE}}_{\mathcal{A}}(\kappa)$ and $\mathsf{Sim}^{\mathcal{FRE}}_{\mathcal{A},\mathcal{S},\mathcal{L}}(\kappa)$, described in Figure 2, are computationally indistinguishable.

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

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

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

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

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

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

Fig. 2. Game defining the $\mathcal{L}$-simulation security of a FRE 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" FRE implies the existence of secure "symmetric" FRE, as stated in Lemma 5.

Lemma 4. *Assuming $\mathcal{FRE}$ is an $\mathcal{L}$-simulation secure function-revealing encryption scheme, then $\mathcal{FRE}$ is an $\mathcal{L}$-indistinguishability secure function-revealing encryption scheme.*

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

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, which is an efficient function-revealing encryption scheme for the function $f_<$ (i.e., order-revealing encryption scheme) with limited leakage, we would like to start with a simple remark. While it seems extremely hard to obtain an $\mathcal{L}_{f_<}$-indistinguishability secure order-revealing encryption scheme from standard assumptions, there is actually a very simple construction that even achieves simulation-based security assuming only one-way functions. However, this construction is only efficient 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 $\{0, \ldots, N-1\} \times \{0, \ldots, 2(N-1)\}$.

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

– $\mathsf{Setup}_<(1^\kappa)$ *picks $K \xleftarrow{\$} \{0,1\}^\kappa$ at random and returns it as the secret key* sk*;*

– $\mathsf{Enc}_<(i, \mathsf{sk}, x)$ *with* $x \in \{0, \ldots, N-1\}$ *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 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* ct_1 *and* 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, "≥", 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, e.g. AES), 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:

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

with $\mathcal{L}_=(\vec{x}_1, \vec{x}_2) = (\mathbb{1}_=(x_{b,i_b}, x_{b,j_b}))_{i_b, j_b \in [|\vec{x}_b|], b \in \{1,2\}}$ where $\mathbb{1}_=(a, b)$ returns 1 if and only if $a = b$. Precisely, $\mathcal{L}_{f_<}(\vec{x}_1, \vec{x}_2)$ reveals exactly the relative order of messages encrypted with index 1 relatively to messages encrypted with index 2, while $\mathcal{L}_=(\vec{x}_1, \vec{x}_2)$ 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 function-revealing encryption scheme for the function* $f_<$*, for polynomial-size message spaces.*

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

4 Order-Revealing Encryption with Limited Leakage

We now describe how to build an order-revealing encryption scheme (a.k.a. function-revealing encryption scheme for $f_<$) from any function-revealing encryption scheme for $f_\#$. As a preliminary, we explain how one can compare two integers by simply checking the disjointness of two sets.

4.1 From Bitstrings to Sets

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

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

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 Σ^0 and Σ^1 are useful as they allow computing the relative order of two integers [LT05]. More precisely, we have:

Lemma 7. *Let x, y be two integers such that $0 \leq x, y < 2^n$ and viewed as n-bit strings. Then*

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

Proof. 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 $i \neq n-1$, $x_{n-1} \| \ldots \| x_{i+1} = y_{n-1} \| \ldots \| y_{i+1}$. If $i = n-1$ this implies $\{1\} \subseteq \Sigma^0(x) \cap \Sigma^1(y)$, and if $i \neq n-1$ this implies $\{x_{n-1} \| \ldots \| x_{i+1} \| 1\} \subseteq \Sigma^0(x) \cap \Sigma^1(y)$.

Suppose now that $\Sigma^0(x) \cap \Sigma^1(y) \neq \varnothing$. If $\Sigma^0(x) \cap \Sigma^1(y) = \{1\}$ then it is clear that $x < y$. Otherwise, there exists an index $i \in \{0, \ldots, n-2\}$ such that $x_{n-1} \| \ldots \| x_{i+1} \| 1 \in \Sigma^0(x) \cap \Sigma^1(y)$. This means that $x_i = 0$, $y_i = 1$, and $x_{n-1} \| \ldots \| x_{i+1} = y_{n-1} \| \ldots \| y_{i+1}$, 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 $x_{n-1} \| \ldots \| x_{i+1} = y_{n-1} \| \ldots \| y_{i+1}$ as well as $x_{n-1} \| \ldots \| x_{j+1} = y_{n-1} \| \ldots \| y_{j+1}$, 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 $x_{n-1} \| \ldots \| x_{j+1} = y_{n-1} \| \ldots \| y_{j+1}$. This concludes the proof. □

Remark 8. Since the cardinality of the intersection is always 0 or 1, one could also base our order-revealing encryption scheme on any function-revealing scheme for the disjointness (i.e. that only reveals if two sets intersect or not).

4.2 A Generic Transform from $\mathcal{FRE}_{\#}$ to $\mathcal{FRE}_{<}$

Our transform simply relies on the above technique. Let $\mathcal{FRE}_{\#} = (\mathsf{Setup}_{\#}, \mathsf{Enc}_{\#}, \mathsf{Eval}_{f_{\#}})$ be a function-revealing encryption scheme for the function $f_{\#}$. For simplicity, 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. This allows us to assume that the sets encrypted by $\mathcal{FRE}_{\#}$ all have the same size. It is very easy to see that Lemma 7 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.

Construction 2 *We build a function-revealing encryption scheme $\mathcal{FRE}_{<} = (\mathsf{Setup}_{<}, \mathsf{Enc}_{<}, \mathsf{Eval}_{f_{<}})$ for the function $f_{<}$ as follows:*

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

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

- $\mathsf{Eval}_{f_{<}}$ *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 $\mathcal{FRE}_{\#}$ and from Lemma 7.

Security. Security immediately follows from the security of $\mathcal{FRE}_{\#}$ and the leakage is simply the leakage associated of $\mathcal{FRE}_{\#}$ 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 $\mathcal{FRE}_{\#}$ is $\mathcal{L}$-indistinguishability secure. Then, we define the leakage of Construction 2 as:

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

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

Theorem 9. *Assuming there exists an $\mathcal{L}$-indistinguishability secure function-revealing 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 $\mathcal{FRE}_<$ obtained via Construction 2. Then, one can build an adversary $\mathcal{B}$ against the $\mathcal{L}$-indistinguishability security of $\mathcal{FRE}_\#$ as follows: when $\mathcal{A}$ makes a query $(i, x^{(0)}, x^{(1)})$ to the encryption oracle, $\mathcal{B}$ does the following. First, 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 $i = 2$. It then 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 9 follows. □

Remark 10. One could also prove in a very similar manner that the obtained construction is simulation-secure assuming the underlying scheme $\mathcal{FRE}_\#$ is simulation-secure.

4.3 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 function-revealing 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 function-revealing 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 $\langle \vec{\alpha}, \vec{\beta} \rangle = 0$, with $\vec{\alpha} = (\alpha_0, \ldots, \alpha_n)$ and $\vec{\beta} = (1, b, b^2, \ldots, b^n)$. Therefore, this can be tested privately using a function-revealing 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 $\mathcal{FRE}_\perp = (\mathsf{Setup}_\perp, \mathsf{Enc}_\perp, \mathsf{Eval}_{f_\perp})$ be a function-revealing encryption scheme for orthogonality testing.

Construction 3 *We build a function-revealing encryption scheme $\mathcal{FRE}_\# = (\mathsf{Setup}_\#, \mathsf{Enc}_\#, \mathsf{Eval}_{f_\#})$ for the function $f_\#$ as follows:*

- $\mathsf{Setup}_\#$ *takes as input the security parameter κ and outputs* $\mathsf{Setup}_\perp(1^\kappa) = \mathsf{sk}$;
- $\mathsf{Enc}_\#$ *takes as input an index $i \in \{1,2\}$, a secret key* 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 $\mathcal{FRE}_\perp$.

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 $\vec{\mathcal{S}} = (\mathcal{S}_1, \ldots, \mathcal{S}_{q_1})$ and $\vec{\mathcal{T}} = (\mathcal{T}_1, \ldots, \mathcal{T}_{q_2})$ encrypted respectively with index 1 and 2, we define:

$$\mathcal{L}_{\#^*}(\vec{\mathcal{S}}, \vec{\mathcal{T}}) = (\#(\mathcal{I} \cap \mathcal{T}_i))_{\mathcal{I} \in \vec{\mathcal{S}}^\cap, i \in [q_2]} \quad , \tag{2}$$

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

Theorem 11. *Assuming there exists an* $\mathcal{L}_\perp$*-indistinguishability secure function-revealing encryption scheme for orthogonality testing, then there exists an* $\mathcal{L}_{\#^*}$*-indistinguishability secure function-revealing encryption scheme for cardinality of intersection.*

The proof of the above theorem is detailed in Appendix C. 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. A formal proof of the latter statement is also given in Appendix C.

4.4 Orthogonality Testing and Relation with Predicate Encryption

We finally describe how we obtain a function-revealing encryption for orthogonality testing, namely for the function

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

This is the last step in building our efficient order-revealing encryption scheme with limited leakage and from standard assumptions.

The existence of such a scheme is actually implied by the existence of a fully-secure secret-key inner-product encryption scheme, which in particular exists under the DLin assumption [BBS04]; e.g., [KT14]. 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 function-revealing encryption scheme for the function f. A very similar result was already proposed in the case of property-preserving encryption in [AAB+13,CD15]. For completeness, definitions of the DLin assumption and of fully-secure secret-key predicate encryption and inner-product encryption are recalled in Appendix D. In particular, note that by fully-secure, we mean predicate-hiding and attribute-hiding.

Theorem 12. *Let $f : \mathcal{M} \times \mathcal{M} \to \{0,1\}$ be any function. Assuming there exists a fully-secure secret-key predicate encryption scheme for the class of predicates $\mathcal{F}_f = \{f_a : b \in \mathcal{M} \mapsto f(a,b) \in \{0,1\} \mid a \in \mathcal{M}\}$, then there exists an $\mathcal{L}_f$-indistinguishability secure function-revealing 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 function-revealing encryption scheme $(\mathsf{Setup}_f, \mathsf{Enc}_f, \mathsf{Eval}_f)$ for f as follows: Setup_f is the same as 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 ct_1 to decrypt 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 12 follows. □

5 Putting Everything Together

We conclude by assembling all our results and obtain an order-revealing encryption scheme with limited leakage assuming the standard DLin assumption. We denote by $\mathcal{L}_\perp$ the (ideal) leakage of the function $f_\perp$ (so $\mathcal{L}_\perp = \mathcal{L}_{f_\perp}$ in the sense of Definition 3).

Corollary 13. *Assuming* DLin, *there exists an $\mathcal{L}_\perp$-indistinguishability secure function-revealing encryption scheme for orthogonality testing.*

Proof. Corollary 13 follows from the existence of a fully-secure inner-product encryption scheme under the DLin assumption, as in [KT14], and from Theorem 12. □

Corollary 14. *Assuming* DLin, *there exists an $\mathcal{L}_{\#^*}$-indistinguishability secure function-revealing encryption scheme for the function $f_{\#}$.*

Proof. Immediate from Theorem 11 and Corollary 13. □

Corollary 15. *Assuming* DLin, *there exists a $\mathcal{L}_{\mathcal{L}_{\#^*}}$-indistinguishability secure function-revealing encryption scheme for the function $f_<$ (a.k.a. order-revealing encryption scheme).*

Proof. Immediate from Theorem 9 and Corollary 14. □

5.1 Detailed Leakage and How to Reduce It

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 $(x_1, \ldots, x_q)$ encrypted respectively with index 1 and an integer y encrypted with index 2. Then, our encryption scheme reveals the order of x_i relatively to y, for any $1 \leq i \leq q$, as expected, but also the following: For every subset $\mathcal{S}$ of $\{x_1, \ldots, x_q\}$ of size at least 2, it reveals 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. Since, as shown in Lemma 7, there is at most one such prefix, it is sufficient to consider only sets $\mathcal{S}$ of size exactly 2 (as to check any larger set $\mathcal{S} = \{x_1, \ldots, x_q\}$, you can just check every set $\mathcal{S}_j = \{x_1, x_j\}$, $j = 2, \ldots, q_2$; It is easy to see that there is such a prefix for $\mathcal{S}$ and y if and only if there is one for every $\mathcal{S}_j$ and y).

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.

Then, considering a sequence of integers $(x_1, \ldots, x_{q_1})$ encrypted respectively with index 1 and $(y_1, \ldots, y_{q_2})$ encrypted with index 2, our scheme leaks precisely the following information:

1. the order of x_i relatively to y_j, for any $1 \leq i \leq q_1$ and $1 \leq j \leq q_2$;
2. for every $1 \leq j \leq q_2$ and every $\mathcal{S} = \{x_i, x_k\}$ with $1 \leq i, k \leq q_1$, if x_i and y_j have the same most significant differing bit as x_k and y_j, *if and only if* $x_i < y_j$ *and* $x_k < y_j$.

Please note that even if x_i and x_j have many common prefixes, an encryption y_j can only reveal that x_i and x_j have one common prefix (amongst the potential $n-1$ common prefixes for any $x_i \neq x_j$). Also, note that the *leakage is ideal* if there is only one element encrypted at index 1, even for an unbounded number of integers encrypted at index 2. Thus, we have the following corollary.

Corollary 16. *Assuming* DLin*, there exists an order-revealing encryption scheme with ideal leakage for bounded ciphertexts at index* 1 *and unbounded ciphertexts at index* 2*.*

The latter construction simply consists in encrypting N times each plaintext with N independent secret keys, where N is a bound on the number of ciphertexts at index 1.

Possible Security Trade-Off. Since the leakage only depends on the number of common prefixes, a simple way to reduce this leakage is to first encrypt the messages with an order-preserving encryption scheme. This results in eliminating part of the common prefixes and reduce the overall leakage.

Another way to reduce the leakage is to improve the underlying construction of function-revealing 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, thanks to Remark 8, one actually just needs to check for disjointness instead of computing the exact cardinality of the intersection (since it is either 0 or 1 anyway). Thus, 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 a non-empty intersection with $\mathcal{A}$, but the size of the ciphertexts grows exponentially with k, since one needs to compute inner-products of vectors of length $(n+1)^k$ instead of vectors of length $n+1$. However, this leads to a solution with ideal leakage if we set $k = n$ (but then again, we need n to be logarithmic for this construction to be polynomial).

5.2 Comparison with [CLOZ16]

Here we precise in more detail how our scheme competes with the recent one proposed by Cash *et al.* in [CLOZ16].

Efficiency. Using the Kawai-Takashima construction [KT14], which is the state of the art for fully-secure secret-key IPE, to instantiate our construction, we obtain the following parameters.

Table 1. Efficiency Comparison with [CLOZ16] for n-bit plaintexts.

	Groups	$\lvert\mathsf{ct}\rvert$ at index 1	$\lvert\mathsf{ct}\rvert$ at index 2	Complexity	Assumption
This paper	Prime order	$6n \cdot \lvert\mathbb{G}\rvert$	$6n^2 \cdot \lvert\mathbb{G}\rvert$	$6n^2$ pairings	DLin
[CLOZ16]	Prime order	$4n \cdot \lvert\mathbb{G}\rvert$	$4n \cdot \lvert\mathbb{G}\rvert$	n^2 pairings	SXDH

As it appears from the table, our construction is slightly less efficient than the concurrent one proposed by Cash *et al.* Yet, as our construction uses in a black-box way a fully-secure (secret-key)

IPE scheme, it would immediately benefits from improvements of this primitive. In particular, in [OT12], Okamoto and Takashima propose an IPE scheme that achieves a slightly weaker security notion (fully-attribute hiding) with constant-size tokens ($10 \cdot |\mathbb{G}|$), and linear-size ciphertexts ($5n \cdot |\mathbb{G}|$). Achieving constant size for one index is particularly interesting for certain applications, in particular for range queries, since one could encrypt a whole data base with constant-size ciphertexts, and make range queries by sending two ciphertexts encrypted with the other index that define the range. Also, any more efficient construction for constructing the cardinality of intersection would immediately improve our construction.

Leakage. While the scheme in [CLOZ16] is slightly more efficient, our scheme achieves a slightly higher level of security. Precisely, as detailed above, considering a sequence of integers $(x_1, \ldots, x_q)$ encrypted respectively with index 1 and an integer y encrypted with index 2, the only additional leakage our scheme reveals is if, for any $1 \leq i, j \leq q$, x_i and y have the same most significant differing bit as x_j and y *if and only if* $x_i < y$ *and* $x_j < y$. The scheme by Cash *et al.* reveals almost the same information, except that if reveals it even if $x_i > y$ or $x_j > y$. Note that this condition being revealed by the functionality of the scheme, our scheme reveals less information. This difference is due to the asymmetry of our construction.

5.3 Applications

To conclude this paper, we propose two applications of our constructions. In particular, these applications do not suffer much from our additional leakage.

Membership Testing on a Database and Searchable Encryption. Our notion of function-revealing 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.

Similarly, one could associate a vector $\vec{x}$ to a data and perform searchable encryption using our function-revealing encryption scheme for orthogonality (whose leakage is ideal). Doing so, one could query all the data whose tag $\vec{x}$ is orthogonal to some vector $\vec{y}$.

Range Queries. Our notion of function-revealing 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. Due to the form of our leakage, the leaked information is ideal if only one such query is made by the user. Moreover, as explained above, the asymmetry of our construction benefits to this application. In particular, a fully-secure secret-key IPE scheme with constant-size tokens or ciphertexts would imply a very efficient solution for range queries.

References

AAB+13. Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit 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.

AAB+15. Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. On the practical security of inner product functional encryption. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 777–798. Springer, Heidelberg, March / April 2015.

AKSX04. Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong 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.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

BCLO09. Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, Heidelberg, April 2009.

BCO11. Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 578–595. Springer, Heidelberg, August 2011.

BKS16. Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 852–880. Springer, Heidelberg, May 2016.

BLR+15. Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 563–594. Springer, Heidelberg, April 2015.

BNPW16. Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 391–418. Springer, Heidelberg, October / November 2016.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

BW07. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, February 2007.

CD15. Sanjit Chatterjee and M. Prem Laxman Das. Property preserving symmetric encryption revisited. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 658–682. Springer, Heidelberg, November / December 2015.

CFL+16. Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new CLT multilinear map over the integers. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 509–536. Springer, Heidelberg, May 2016.

CGKO06. Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 79–88. ACM Press, October / November 2006.

CHL+15. Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, April 2015.

CLOZ16. David Cash, Feng-Hao Liu, Adam O'Neill, and Cong Zhang. Reducing the leakage in practical order-revealing encryption. Cryptology ePrint Archive, Report 2016/661, 2016. http://eprint.iacr.org/2016/661.

CLWW16. Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. Practical order-revealing encryption with limited leakage. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 474–493. Springer, Heidelberg, March 2016.

DDC16. F. Betül Durak, Thomas M. DuBuisson, and David Cash. What else is revealed by order-revealing encryption? In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1155–1166. ACM Press, October 2016.

GGG+14. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.

GKP+13. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.

HJL+17. Helene Haagh, Yue Ji, Chenxing Li, Claudio Orlandi, and Yifan Song. Revealing encryption for partial ordering. In Máire O'Neill, editor, *16th IMA International Conference on Cryptography and Coding*, volume 10655 of *LNCS*, pages 3–22. Springer, Heidelberg, December 2017.

KS17. Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 122–151. Springer, Heidelberg, May 2017.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008.

KT14. Yutaka Kawai and Katsuyuki Takashima. Predicate- and attribute-hiding inner product encryption in a public key setting. In Zhenfu Cao and Fangguo Zhang, editors, *PAIRING 2013*, volume 8365 of *LNCS*, pages 113–130. Springer, Heidelberg, November 2014.

LT05. Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 456–466. Springer, Heidelberg, June 2005.

LW16. Kevin Lewi and David J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1167–1178. ACM Press, October 2016.

OT12. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 591–608. Springer, Heidelberg, April 2012.

PR12. Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 375–391. Springer, Heidelberg, April 2012.

SSW09. Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473. Springer, Heidelberg, March 2009.

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

A Proofs of Lemmata in Section 2.3

A.1 Proof of Lemma 4

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

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

A.2 Proof of Lemma 5

Indistinguishability Security. Let $\mathcal{FRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$ be an $\mathcal{L}$-indistinguishability secure "asymmetric" function-revealing encryption scheme for a k-ary function f. Then, $\mathcal{FRE}' = (\mathsf{Setup}, \mathsf{Enc}', \mathsf{Eval}_f)$ with $\mathsf{Setup}, \mathsf{Eval}_f$ being the same as in $\mathcal{FRE}$ 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" function-revealing encryption

scheme for the function f. The proof is immediate: given an adversary $\mathcal{A}$ against the indistinguishability security of $\mathcal{FRE}'$ 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 $\mathcal{FRE}$ 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 $\mathsf{sym}_{\mathcal{L}}(\vec{x}^{(0)}) = \mathsf{sym}_{\mathcal{L}}(\vec{x}^{(1)})$.

Simulation Security. Let $\mathcal{FRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_f)$ be an $\mathcal{L}$-simulation secure "asymmetric" function-revealing encryption scheme for a k-ary function f. Then, $\mathcal{FRE}' = (\mathsf{Setup}, \mathsf{Enc}', \mathsf{Eval}_f)$ with $\mathsf{Setup}, \mathsf{Eval}_f$ being the same as in $\mathcal{FRE}$ 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 function-revealing encryption scheme for the function f. The proof is immediate: given an adversary $\mathcal{A}$ against the $\mathsf{sym}_{\mathcal{L}}$-simulation security of $\mathcal{FRE}'$ that makes a sequence of queries $(x_1, \ldots, x_q)$, one can simply build an adversary $\mathcal{B}$ against the simulation security of $\mathcal{FRE}$ 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 $\mathcal{FRE}$ guarantees that the distribution of ciphertexts obtained is indistinguishable from the one computed by the simulator $\mathcal{S}$ given only $\mathcal{L}(\vec{x}, \ldots, \vec{x})$.

Lemma 5 easily follows. □

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 $\mathcal{FRE}_<$, 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 $\mathsf{Real}_{\mathcal{A}}^{\mathcal{FRE}_<}$ and $\mathsf{Sim}_{\mathcal{A},\mathcal{S},\mathcal{L}_{<,=}}^{\mathcal{FRE}_<}$ are computationally indistinguishable.

The distribution output by experiment $\mathsf{Real}_{\mathcal{A}}^{\mathcal{FRE}_<}$ consists in a sequence of ciphertexts $(\vec{\mathsf{ct}}_1, \vec{\mathsf{ct}}_2)$ 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 $\mathsf{Real}_{\mathcal{A}}^{\mathcal{FRE}_<}$ is computationally indistinguishable from the distribution 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 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 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 makes a query $(1,x_t)$ and let $\vec{x}_1 = \vec{x}_1.\mathsf{append}(x_t)$. $\mathcal{S}_t$ is executed on input $\mathsf{st}_{\mathcal{S}}$ and $\mathcal{L}_{<,=}(\vec{x}_1,\vec{x}_2)$. 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 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 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 $\vec{x}_2 = \vec{x}_2.\mathsf{append}(x_t)$. $\mathcal{S}_t$ is executed on input $\mathsf{st}_{\mathcal{S}}$ and $\mathcal{L}_{<,=}(\vec{x}_1,\vec{x}_2)$. 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 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 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 Hyb, which is computationally indistinguishable from the distribution of $\mathsf{Real}_{\mathcal{A}}^{\mathcal{FRE}_<}$. Theorem 6 follows. □

C Proof of Theorem 11

First, let us prove 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. This can be done as follows: Consider a sequence of queries $(1,\mathcal{S}_1^{(0)},\mathcal{S}_1^{(1)}),\ldots,(1,\mathcal{S}_{q_1}^{(0)},\mathcal{S}_{q_1}^{(1)}),(2,\mathcal{T}_1^{(0)},\mathcal{T}_1^{(1)}),\ldots,(2,\mathcal{T}_{q_2}^{(0)},\mathcal{T}_{q_2}^{(1)})$, simply denoted as $\boldsymbol{Q}$ in what follows. We recall that we assume every set to be of fixed size n. We define $\mathsf{isValid}(\boldsymbol{Q})$ as the predicate $\mathcal{L}_{\#^*}(\vec{\mathcal{S}}^{(0)},\vec{\mathcal{T}}^{(0)}) = \mathcal{L}_{\#^*}(\vec{\mathcal{S}}^{(1)},\vec{\mathcal{T}}^{(1)})$, so that $\mathsf{isValid}(\boldsymbol{Q}) = 1$ if and only if the sequence of queries $\boldsymbol{Q}$ is valid.

Computing $\mathsf{isValid}(\boldsymbol{Q})$ can be done in polynomial-time by the following algorithm:

1. Build the $2 \cdot q_2$ tables $\mathsf{U}_\ell^{(0)},\mathsf{U}_\ell^{(1)} \in \{0,1\}^{n \times q_1}$, $\ell = 1,\ldots,q_2$, defined as $\mathsf{U}_\ell^{(b)}[i,j] = 1$ if and only if $t_{\ell,i}^{(b)} \in \mathcal{S}_j^{(b)}$, where $\{t_{\ell,1}^{(b)},\ldots,t_{\ell,n}^{(b)}\} = \mathcal{T}_\ell^{(b)}$. Note that the order of the rows (i.e. the order of the elements in $\mathcal{T}_\ell^{(b)}$) is arbitrary but the order of columns is fixed by the order of the queries (column j corresponds to j-th query $(1,\mathcal{S}_j^{(0)},\mathcal{S}_j^{(1)})$).
2. For every $1 \le \ell \le q_2$ and $b \in \{0,1\}$, consider rows of $\mathsf{U}_\ell^{(b)}$ as q_1-bit integers and re-order the rows of $\mathsf{U}_\ell^{(b)}$ according to the lexicographic order of these integers (say in decreasing order). Denote $\mathsf{ord}.\mathsf{U}_\ell^{(b)}$ the resulting table.

This process is clearly polynomial-time as soon as q_1 and q_2 are polynomial, which directly follows from the fact that the adversary is polynomial-time.

Claim. $\mathsf{isValid}(\boldsymbol{Q}) = 1$ if and only if $\mathsf{ord}.\mathsf{U}_\ell^{(0)} = \mathsf{ord}.\mathsf{U}_\ell^{(1)}$ for all $1 \le \ell \le q_2$.

Proof. The first direction is immediate, as if $\mathsf{isValid}(\boldsymbol{Q}) = 1$, it is clear that $\mathsf{ord.U}_{\ell}^{(0)} = \mathsf{ord.U}_{\ell}^{(1)}$ for all $1 \leq \ell \leq q_2$.

Let us prove the more delicate second direction. We assume without loss of generality that $q_2 = 1$. We thus simplify notations by removing the index ℓ and we denote by $(2, \mathcal{T}^{(0)}, \mathcal{T}^{(1)})$ the single query made at index 2. Let us denote by $k+1$ the index of the first column in which $\mathsf{ord.U}^{(0)}$ and $\mathsf{ord.U}^{(1)}$ differ and denote $\mathsf{ord.U.k}^{(b)}$ the subtable obtained by taking only the first $k+1$ columns of $\mathsf{ord.U}^{(b)}$, for $b \in \{0,1\}$. Each row of $\mathsf{ord.U.k}^{(b)}$ can be seen as a k-bit integer and rows of table $\mathsf{ord.U}^{(b)}$ being ordered, so are rows of table $\mathsf{ord.U.k}^{(b)}$. We regroup rows of $\mathsf{ord.U.k}^{(b)}$ into blocks of consecutive rows that have the same k-bit prefix. For any prefix $p \in \{0,1\}^k$, we associate the block $\mathsf{B}_p^{(b)}$ of consecutive rows of $\mathsf{ord.U.k}^{(b)}$ that start with prefix p. Then, for each corresponding block, we just need to prove that there is the same number of 1 in the $(k+1)$-th column of $\mathsf{B}_p^{(0)}$ and $\mathsf{B}_p^{(1)}$ for contradiction (the number of 0 then being the same as well since both blocks have the same size as $\mathsf{ord.U}^{(b)}$ do not differ on their first k columns).

We show by induction that the number of 1 in the last column of a block can be expressed as a sum of cardinal of intersections (of the form considered in the definition of $\mathcal{L}_{\#^*}$ –see Equation 2–) and thus are equal by definition of queries being valid.

Assume the rows are ordered in decreasing order. Then, the first block is indexed by prefix 1^k.

- Base case: The number of 1 in the last column of block $\mathsf{B}_{1^k}^{(b)}$ corresponds precisely to $\#(\mathcal{T}^{(b)} \cap (\bigcap_{j=1}^{k+1} \mathcal{S}_j^{(b)}))$, then the claim holds.
- Induction: Let $p \in \{0,1\}^k$. The number of 1 in block $\mathsf{B}_p^{(b)}$ can be expressed as $\#(\mathcal{T}^{(b)} \cap (\bigcap_{j=1|p_j=1}^{k+1} \mathcal{S}_j^{(b)})) - (\sum_{p' \geq p} \#(\mathcal{T}^{(b)} \cap (\bigcap_{j=1|p'_j=1}^{k+1} \mathcal{S}_j^{(b)})))$ where $p' \geq p$ if and only if $p'_j \geq p_j$ for all $1 \leq j \leq k+1$. The terms of the latter sum correspond to all the elements of $\mathcal{T}^{(b)} \cap (\bigcap_{j=1|p_j=1}^{k+1} \mathcal{S}_j^{(b)})$ that were already counted as being elements of an intersection of the these sets with some more sets. By recursion, all the terms of the latter sum can be expressed by a sum of cardinal of intersections (of the form considered in the definition of $\mathcal{L}_{\#^*}$) as $p' \geq p \Rightarrow p' > p$. Hence, they are equal by definition of queries being valid.

Therefore, the $(k+1)$-th columns of $\mathsf{ord.U}^{(0)}$ and $\mathsf{ord.U}^{(1)}$ are equal, which concludes the proof by contradiction. □

Now that it is clear that the challenger can check the validity of queries made by an adversary in polynomial-time, let us prove Theorem 11.

Proof. Let $\mathcal{A}$ be an adversary against the $\mathcal{L}_{\#^*}$-indistinguishability security of the scheme $\mathcal{FRE}_{\#} = (\mathsf{Setup}_{\#}, \mathsf{Enc}_{\#}, \mathsf{Eval}_{f_{\#}})$ obtained via Construction 3, 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 $\mathcal{FRE}_{\perp} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}_{f_{\perp}})$ 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 list_0 and $\mathcal{S}^{(1)}$ to list_1, then simply computes $\vec{\alpha}^{(0)} = \mathsf{coef}(\mathcal{S}^{(0)})$ and $\vec{\alpha}^{(1)} = \mathsf{coef}(\mathcal{S}^{(1)})$, queries $(1, \vec{\alpha}^{(0)}, \vec{\alpha}^{(1)})$ 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 out and applies the following process, termed 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 $\vec{\beta}^{(0)} = \mathsf{exp}(s^{(0)})$ and $\vec{\beta}^{(1)} = \mathsf{exp}(s^{(1)})$, and queries $(2, \vec{\beta}^{(0)}, \vec{\beta}^{(1)})$ to its encryption oracle and adds the value it gets to the list out. It then reiterates 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 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 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 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 $\mathcal{L}_{\#^*}(\vec{\mathcal{S}}_1^{(0)}, \vec{\mathcal{S}}_2^{(0)}) = \mathcal{L}_{\#^*}(\vec{\mathcal{S}}_1^{(1)}, \vec{\mathcal{S}}_2^{(1)})$, where $\vec{\mathcal{S}}_i^{(b)}$ denote the series of left (if $b = 0$) or right (if $b = 1$) queries made at index i. This implies directly that Pair always terminates.

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

Finally, one needs to show that $\mathcal{B}$ simulates correctly the oracle, which is immediate from the description of Construction 3. This concludes the proof of Theorem 11. □

D Additional Definitions

D.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 $a_i, w_i \xleftarrow{\$} \mathbb{Z}_N$ for $i = 1, 2$. The DLin assumption corresponds to the hardness of the DLin problem.

D.2 Predicate Encryption and Inner Product Encryption

Definition 17 (Secret-Key Predicate Encryption). *A* secret-key predicate encryption *scheme is a tuple of PPT algorithms* (Setup, TokenGen, Enc, Dec)*, defined as follows:*

- Setup *takes as input the security parameter* 1^κ *and ouputs a secret key* sk *and public parameters* pp*;*
- TokenGen *takes as inputs a secret key* sk *and a predicate* $\vec{P}$ *and outputs a token* tk_P*;*
- Enc *takes as inputs a secret key* sk *and an attribute* I *and a message* x *and outputs a ciphertext* $\mathsf{ct}_{I,x}$*;*
- Dec *takes as input a token* tk_P *and a ciphertext* $\mathsf{ct}_{I,x}$ *and outputs* x *or* $\perp$*.*

For correctness, we require that for any $\mathsf{sk} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$ *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 tk (resp. a ciphertext 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 tk_{P_b} (resp. ciphertext 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 18 (Secret-Key Inner Product Encryption). *A* secret-key inner product encryption *scheme is a tuple of probabilistic polynomial-time algorithms* (Setup, TokenGen, Enc, Query)*, defined as follows:*

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

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

Security. A secret-key inner product encryption scheme is fully-secure if a token tk (resp. a ciphertext 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 $(\vec{y}_0, \vec{y}_1)$ (resp. pair of attributes $(\vec{x}_0, \vec{x}_1)$) to get token $\mathsf{tk}_{\vec{y}_b}$ (resp. ciphertext $\mathsf{ct}_{\vec{x}_b}$), 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 $(\vec{x}_0^{(1)}, \vec{x}_1^{(1)}), \ldots, (\vec{x}_0^{(q_1)}, \vec{x}_1^{(q_1)}), (\vec{y}_0^{(1)}, \vec{y}_1^{(1)}), \ldots, (\vec{y}_0^{(q_2)}, \vec{y}_1^{(q_2)})$, we require that for all $1 \leq i \leq q_1$ and $1 \leq j \leq q_2$, $\langle \vec{x}_0^{(i)}, \vec{y}_0^{(j)} \rangle = 0 \iff \langle \vec{x}_1^{(i)}, \vec{y}_1^{(j)} \rangle = 0$.