# Multi-key and Multi-input Predicate Encryption from Learning with Errors

Danilo Francati[1](✉), Daniele Friolo[2], Giulio Malavolta[3], and Daniele Venturi[2]

[1] Aarhus University, Aarhus, Denmark
dfrancati@cs.au.dk
[2] Sapienza University of Rome, Rome, Italy
[3] Max Planck Institute for Security and Privacy, Bochum, Germany

**Abstract.** We put forward two natural generalizations of predicate encryption (PE), dubbed *multi-key* and *multi-input* PE. More in details, our contributions are threefold.

- **Definitions.** We formalize security of multi-key PE and multi-input PE following the standard indistinguishability paradigm, and modeling security both against malicious senders (i.e., corruption of encryption keys) and malicious receivers (i.e., collusions).
- **Constructions.** We construct adaptively secure multi-key and multi-input PE supporting the conjunction of poly-many arbitrary single-input predicates, assuming the sub-exponential hardness of the learning with errors (LWE) problem.
- **Applications.** We show that multi-key and multi-input PE for expressive enough predicates suffices for interesting cryptographic applications, including non-interactive multi-party computation (NI-MPC) and matchmaking encryption (ME).

In particular, plugging in our constructions of multi-key and multi-input PE, under the sub-exponential LWE assumption, we obtain the first ME supporting *arbitrary policies* with unbounded collusions, as well as robust (resp. non-robust) NI-MPC for so-called *all-or-nothing* functions satisfying a non-trivial notion of reusability and supporting a constant (resp. polynomial) number of parties. Prior to our work, both of these applications required much heavier tools such as indistinguishability obfuscation or compact functional encryption.

**Keywords:** predicate encryption · non-interactive MPC · matchmaking encryption · LWE

## 1 Introduction

Predicate encryption (PE) [17,30,37] is a powerful cryptographic primitive that enriches standard encryption with fine-grained access control to the encrypted data. In PE, the ciphertext is associated to both a message $m$ and an attribute[1]

[1] Sometimes, we also refer to $x$ as the predicate input. Throughout the paper, we use the terms attribute and input interchangeably.

$x$, whereas the secret key is associated to a predicate $\mathbb{P}$, in such a way that the decryption process reveals the message if and only if the attribute $x$ satisfies the predicate $\mathbb{P}$ (i.e., $\mathbb{P}(x) = 1$). Typically, security of PE requires indistinguishability in the presence of *collusion attacks*, namely, for any pair of attributes $(x^0, x^1)$ and for any pair of messages $(m^0, m^1)$, ciphertexts corresponding to $(x^0, m^0)$ and to $(x^1, m^1)$ are computationally indistinguishable, even for an adversary possessing poly-many decryption keys $\mathsf{dk}_{\mathbb{P}}$, so long as $\mathbb{P}(x^0) = \mathbb{P}(x^1) = 0$ (otherwise it is easy to distinguish).

Recently, there has been a lot of progress in constructing PE supporting expressive predicates under standard assumptions [5,12,17,30,37,38,42,43,45,46]. In particular, Gourbunov, Vaikuntanathan and Wee [30] give a construction of selectively secure PE (with unbounded collusions) for arbitrary predicates under the learning with errors (LWE) assumption. Moreover, under subexponential LWE, the same construction achieves adaptive security (this requires complexity leveraging).

## 1.1 Our Contributions

In this paper, we put forward two natural generalizations of PE which we dub *multi-key* PE and *multi-input* PE. Furthermore, we construct both multi-key PE and multi-input PE for a particular class of predicates, under the LWE assumption. As we show, the class of predicates our schemes can handle is powerful enough to yield interesting cryptographic applications, including matchmaking encryption (ME) [10,11] for arbitrary policies and non-interactive multi-party computation (NI-MPC) [34] satisfying a weaker (but still non-trivial) notion of reusability. We elaborate on these contributions in Sect. 1.3.

Prior to our work, all of the above applications required much stronger tools such as indistinguishability obfuscation (iO) [13]. While recent work made significant progress towards basing iO on standard assumptions [35,36], these constructions are fairly complex and still require a careful combination of multiple assumptions (i.e., learning parity with noise, the SXDH assumption on bilinear groups, and the existence of pseudorandom generators computable in constant depth). Furthermore, such constructions are not secure in the presence of a quantum attacker. Candidate constructions of *post-quantum* iO also exist [18,28,47], but they are based on problems whose hardness is less understood.

*Multi-key PE.* In multi-key PE, we consider an ensemble of predicates $\mathcal{P} = \{\mathbb{P}_v\}$ indexed by a value $v \in \mathcal{V}$ which is uniquely represented as a sequence $v = (v_1, \ldots, v_n) \in \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$. A sender can encrypt a message under an input $x$ using the public-key encryption algorithm $\mathsf{Enc}(\mathsf{mpk}, x, m)$. A trusted authority generates decryption keys $\mathsf{dk}_{v_i}$ (using the corresponding master secret key $\mathsf{msk}_i$) for each $i \in [n]$, with the guarantee that, given the decryption keys $\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}$, the receiver can decrypt successfully the ciphertext $c$ (associated to plaintext $m$ and attributes $x$), so long as $\mathbb{P}_v(x) = \mathbb{P}_{v_1,\ldots,v_n}(x) = 1$.

Security of multi-key PE says that, for any pair of attributes $(x^0, x^1)$ and for any pair of messages $(m^0, m^1)$, ciphertexts $c$ associated to $(x^0, m^0)$ and $(x^1, m^1)$

should be computationally indistinguishable even under unbounded collusions, where the latter essentially means that the adversary can obtain decryption keys for (poly-many) arbitrary values $v_1, \dots, v_n$ which correspond to predicates indexed by any value $v = (v_1, \dots, v_n)$ such that $\mathbb{P}_v(x^0) = \mathbb{P}_v(x^1) = 0$. This yields so-called CPA-1-sided security. The stronger notion of CPA-2-sided security additionally allows for predicates indexed by values $v$ such that $\mathbb{P}_v(x^0) = \mathbb{P}_v(x^1) = 1$, so long as $m^0 = m^1$. These notions mimic the corresponding notions that are already established for standard PE.

Our first result is a construction of multi-key PE, from the sub-exponential LWE assumption, supporting conjunctions of arbitrary predicates, i.e. for predicates of the form $\mathbb{P}_v(x) = \mathbb{P}_{v_1}(x_1) \wedge \dots \wedge \mathbb{P}_{v_n}(x_n)$, where $x = (x_1, \dots, x_n)$ and $v = (v_1, \dots, v_n)$.

**Theorem 1 (Informal).** *Assuming the sub-exponential hardness of LWE, there exists a CPA-1-sided adaptively secure multi-key PE scheme supporting conjunctions of $n = \mathsf{poly}(\lambda)$ arbitrary predicates with unbounded collusions.*

*Multi-input PE.* In multi-input PE, we consider predicates $\mathbb{P}$ with $n$ inputs, i.e. predicates of the form $\mathbb{P}(x_1, \dots, x_n)$. A trusted authority produces encryption keys $\mathsf{ek}_i$ which are associated to the $i$-th slot of an input for $\mathbb{P}$; namely, given a (possibly secret)[2] encryption key $\mathsf{ek}_i$, a sender can generate a ciphertext $c_i$ which is an encryption of message $m_i$ under attribute $x_i$. At the same time, the authority can produce a decryption key $\mathsf{dk}_{\mathbb{P}}$ associated to an $n$-input predicate $\mathbb{P}$, with the guarantee that the receiver can successfully decrypt $c_1, \dots, c_n$, and thus obtain $m_1, \dots, m_n$, so long as $\mathbb{P}(x_1, \dots, x_n) = 1$.

As for security, we consider similar flavors as CPA-1-sided and CPA-2-sided security for standard PE. Namely, for any pair of sequences of attributes $(x_1^0, \dots, x_n^0)$ and $(x_1^1, \dots, x_n^1)$ and for any pair of sequences of messages $(m_1^0, \dots, m_n^0)$ and $(m_1^1, \dots, m_n^1)$, ciphertexts $c_1, \dots, c_n$ corresponding to either $(x_1^0, m_1^0), \dots, (x_n^0, m_n^0)$ or $(x_1^1, m_1^1), \dots, (x_n^1, m_n^1)$ should be computationally indistinguishable. Here, we additionally consider two cases:

- In the setting with no corruptions (a.k.a. the secret-key setting), all of the encryption keys $\mathsf{ek}_i$ are secret and cannot be corrupted (and thus all the senders are honest).
- In the setting with adaptive corruptions, the attacker can adaptively reveal some of the encryption keys $\mathsf{ek}_i$ (and thus corrupt a subset of the senders).

Naturally, for both of these flavors, one can define CPA-1-sided and CPA-2-sided security with or without collusions.

Our second result is a construction of multi-input PE, from the sub-exponential LWE assumption, supporting conjunctions of $n = \mathsf{poly}(\lambda)$ arbitrary predicates *with wildcards*, i.e. for predicates of the form $\mathbb{P}(x_1, \dots, x_n) =$

[2] This is one of the differences between multi-key PE and multi-input PE: the former has a public-key encryption algorithm, whereas the latter could have a secret-key encryption algorithm.

$\mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)$ such that, for each $i \in [n]$, there exists a (public) wildcard input $x_i^\star$ for which $\mathbb{P}_i(x_i^\star) = 1$ for every $i$-th predicate $\mathbb{P}_i$.[3] Our multi-input PE construction retains its security only in the setting of no corruptions (i.e., the encryption keys $\mathsf{ek}_i$ are kept secret) and no collusions (i.e., the adversary only knows a single decryption key $\mathsf{dk}_\mathbb{P}$ for an adversarially chosen predicate $\mathbb{P}$).

**Theorem 2 (Informal).** *Assuming the sub-exponential hardness of LWE, there exists a CPA-1-sided adaptively secure multi-input PE scheme supporting conjunctions of $n = \mathsf{poly}(\lambda)$ arbitrary predicates with wildcards, without corruptions and without collusions.*

Our third result is a construction of multi-input PE, from the sub-exponential LWE assumption, supporting the same class of predicates as above but tolerating adaptive corruptions of up to $n-1$ parties. However, this particular scheme only supports predicates with constant arity.

**Theorem 3 (Informal).** *Assuming the sub-exponential hardness of LWE, there exists a CPA-1-sided adaptively secure multi-input PE scheme supporting conjunctions of $n = O(1)$ arbitrary predicates with wildcards, under $n-1$ adaptive corruptions and without collusions.*

Finally, we anticipate that all our constructions are transformations that leverage single-input PE schemes (e.g., [30]) and lockable obfuscation [31,48] as building blocks. Such transformations are general and achieve CPA-2-sided security if the underlying single-input PE schemes are CPA-2-sided secure. In particular, we obtain $(i)$ CPA-2-sided secure multi-key PE with unbounded collusions for $n = \mathsf{poly}(\lambda)$, $(ii)$ CPA-2-sided secure multi-input PE without corruptions and without collusions for $n = O(\log(\lambda))$,[4] and $(iii)$ CPA-2-sided secure multi-input PE under $n-1$ corruptions and without collusions for $n = O(1)$. However, at the time of this writing, the LWE assumption is not sufficient for CPA-2-sided security. Indeed, even for single-input PE for arbitrary predicates, CPA-2-sided security implies iO [15]. The current state-of-the-art constructions of iO require much stronger assumptions compared to standard LWE.

## 1.2 Technical Overview

We now give a high level overview of our constructions. As explained above, both our multi-key and multi-input PE constructions handle conjunctions of arbitrary predicates, i.e., predicates of the form:

$$\mathbb{P}(x_1, \ldots, x_n) = \mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n). \tag{1}$$

[3] Note that, in the setting with no corruptions, assuming the presence of a (single) wildcard $x_i^\star$ for each $\mathbb{P}_i$ does not affect the expressiveness and the security guarantees of multi-input PE. This is because the $i$-th sender can simply choose not to encrypt $x_i^\star$, which will not permit the receiver to evaluate $\mathbb{P}_i$ over $x_i^\star$.

[4] Note that, in case of no corruptions, our CPA-1-sided construction supports $n = \mathsf{poly}(\lambda)$. However, to achieve CPA-2-sided security we use complexity leveraging and this reduces $n$ from $\mathsf{poly}(\lambda)$ to $O(\log(\lambda))$.

We start by explaining how to build multi-key PE for the above class of predicates by combining single-input PE and so-called lockable obfuscation [31,48]. Informally, a lockable obfuscation scheme allows to obfuscate a circuit $\mathbb{C}$ under a lock $y$ together with a message $m$, in such a way that evaluating the obfuscated circuit, on input $x$, returns $m$ if $\mathbb{C}(x) = y$. As for security, an obfuscated circuit can be simulated in a virtual black-box (VBB) fashion whenever the lock is random and unknown to the adversary. Lockable obfuscation exists under the standard LWE assumption.

Then, we explain how to build multi-input PE (for the same class of predicates) by additionally using SKE and PKE. Here, we consider two settings: without corruptions (a.k.a. the secret-key setting) and with corruptions. The former assumes that all the encryption keys (each corresponding to an input) are secret. The latter is a stronger model that allows the adversary to leak one or more encryption keys (i.e., corruption of the senders). We achieve security in each setting by changing the way lockable obfuscation is used. In particular, part of the contribution of this paper is a new technique based on nested (lockable obfuscated) circuits that execute each other. This technique allows us to construct a multi-input PE that can handle adaptive corruptions. We provide a high-level overview in the remaining part of this section. For more details, we refer the reader to Sect. 4, Sect. 5, and the full version of this work [25].

*Multi-key Predicate Encryption.* An $n$-key PE allows a sender to encrypt a message $m$ under an attribute $x$, by running $c \leftarrow\$ \mathsf{Enc}(\mathsf{mpk}, x, m)$. Similarly to single-input PE, a receiver can correctly decrypt $c$ if it has a decryption key for a predicate $\mathbb{P}_v$, within a family $\mathcal{P}$ of predicates indexed by values $v \in \mathcal{V}$, such that $\mathbb{P}_v(x) = 1$. The main difference between single-input PE and $n$-key PE is that in the latter the receiver must have $n$ independent decryption keys $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$ that uniquely represent the predicate $\mathbb{P}_v(\cdot) = \mathbb{P}_{v_1,\ldots,v_n}(\cdot)$, i.e., the decryption key associated to a particular predicate is decomposed into $n$ decryption keys. Each decryption key $\mathsf{dk}_{v_i}$ is generated by the authority via $\mathsf{KGen}(\mathsf{msk}_i, v_i)$ where $(\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$ are the master secret keys generated during the setup. Hence, once obtained $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$ from the authority, the receiver can decrypt the ciphertext $c$ (encrypted under attribute $x$) by executing $\mathsf{Dec}(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}, c)$. The message is returned if the predicate $\mathbb{P}_{v_1,\ldots,v_n}(x) = 1$, where $\mathbb{P}_{v_1,\ldots,v_n}(\cdot)$ is the predicate represented by the combination of the $n$ decryptions keys $\mathsf{dk}_{v_1}, \ldots,$ $\mathsf{dk}_{v_n}$. The security of $n$-key PE is analogous to that of single-input PE, where the validity of the adversary $\mathsf{A}$ is defined with respect to the (poly-many) tuples $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$ of $n$ decryption keys that the adversary has access to. In particular, we consider the well-known notion of CPA-1-sided security, i.e., the attacker cannot distinguish between $\mathsf{Enc}(\mathsf{mpk}, x^0, m^0)$ and $\mathsf{Enc}(\mathsf{mpk}, x^1, m^1)$ so long as it only holds combinations of $n$ decryption keys $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$ such that $\mathbb{P}_{v_1,\ldots,v_n}(x^0) = \mathbb{P}_{v_1,\ldots,v_n}(x^1) = 0$ (i.e., the adversary cannot decrypt the challenge ciphertext).[5]

As explained above, we focus on conjunctions of arbitrary predicates $\mathbb{P}_{v_1,\ldots,v_n}(x) = \mathbb{P}_{v_1,\ldots,v_n}(x_1,\ldots,x_n) = \mathbb{P}_{v_1}(x_1) \wedge \cdots \wedge \mathbb{P}_{v_n}(x_n)$ as defined in Eq. (1); hence, $x = (x_1,\ldots,x_n)$ and each $\mathsf{dk}_{v_i}$ identifies the $i$-th predicate of the conjunction (and, in turn, any tuple of $n$ decryption keys uniquely identifies the global predicate). We build an $n$-key PE handling this class of predicates by extending the technique of Goyal et al. [31], that uses lockable obfuscation to transform any CPA secure attribute-based encryption (ABE) (recall that ABE schemes only guarantee the secrecy of the message) into a CPA-1-sided secure PE (i.e., secrecy of both message and attribute). Let $\mathsf{PE}_i = (\mathsf{Setup}_i, \mathsf{KGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ for $i \in [n]$ be $n$ single-input PE schemes, each with ciphertext expansion $\mathsf{poly}(\lambda) + |m_i|$ where $|m_i|$ is the message length supported by the $i$-th PE.[6] In a nutshell, our $n$-key PE scheme $\mathsf{kPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ works as follows. The setup algorithm $\mathsf{Setup}$ simply executes $\mathsf{Setup}_i$ of each $\mathsf{PE}_i$ and outputs the master public key $\mathsf{mpk} = (\mathsf{mpk}_1,\ldots,\mathsf{mpk}_n)$ and $n$ master secret keys $(\mathsf{msk}_1,\ldots,\mathsf{msk}_n)$. To generate a decryption key $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, v_i)$ (representing the $i$-th predicate $\mathbb{P}_{v_i}(\cdot)$ of the conjunction), the authority can use the key generation algorithm of the $i$-th PE, i.e., $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_i, \mathbb{P}_{v_i})$. To encrypt a message $m$ under an input $x = (x_1,\ldots,x_n)$, a sender samples a random lock $y$ and encrypts it $n$ times using $\mathsf{PE}_1,\ldots,\mathsf{PE}_n$, i.e., $c \leftarrow\!\!\$\ \mathsf{Enc}_n(\mathsf{mpk}_n, x_n, \mathsf{Enc}_{n-1}(\mathsf{mpk}_{n-1}, x_{n-1}, \cdots, \mathsf{Enc}_1(\mathsf{mpk}_1, x_1, y)))$. Note that, for $n = \mathsf{poly}(\lambda)$, the final ciphertext will be of polynomial size since each underlying $i$-th PE scheme has $\mathsf{poly}(\lambda) + |m_i|$ ciphertext expansion where $|m_i|$ is the message length supported by $i$-th scheme.

The final ciphertext of the $n$-key PE $\mathsf{kPE}$ will be the obfuscation of the circuit $\mathbb{C}_c$ under the lock $y$ together with the message $m$ (i.e., $\widetilde{\mathbb{C}} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_c, y, m)$), where $\mathbb{C}_c$, on input $(\mathsf{dk}_{v_1},\ldots,\mathsf{dk}_{v_n})$, iteratively decrypts $c$ and returns the last decrypted value, i.e., $y = \mathbb{C}_c(\mathsf{dk}_{v_1},\ldots,\mathsf{dk}_{v_n}) = \mathsf{Dec}_1(\mathsf{dk}_{v_1}, \cdots, \mathsf{Dec}_n(\mathsf{dk}_{v_n}, c))$. Decryption is straightforward: the receiver simply executes $\widetilde{\mathbb{C}}$ using its $n$ decryption keys.

The CPA-1-sided security of our construction follows by the CPA security (i.e., secrecy of the message) of $\mathsf{PE}_1,\ldots,\mathsf{PE}_n$ and by the security of lockable obfuscation.[7] Intuitively, the proof works as follows. In order to be valid, an adversary $\mathsf{A}$ cannot hold a tuple of decryption keys $(\mathsf{dk}_{v_1},\ldots,\mathsf{dk}_{v_n})$ such that $\mathbb{P}_{v_1,\ldots,v_n}(x^b) = \mathbb{P}_{v_1,\ldots,v_n}(x_1^b,\ldots,x_n^b) = 1$, where $x^b = (x_1^b,\ldots,x_n^b)$ is the

[5] Observe that the decryption keys can be interleaved. For example, starting from $(\mathsf{dk}_{v_1},\ldots,\mathsf{dk}_{v_i},\ldots\mathsf{dk}_{v_n})$ representing the predicate $\mathbb{P}_{v_1,\ldots,v_i,\ldots,v_n}$, the adversary can ask for an additional $i$-th decryption key $\mathsf{dk}_{v_i'}$ and rearrange the decryption keys as $(\mathsf{dk}_{v_1},\ldots,\mathsf{dk}_{v_i'},\ldots\mathsf{dk}_{v_n})$ in order to obtain the tuple representing a different predicate $\mathbb{P}_{v_1,\ldots,v_i',\ldots,v_n} \neq \mathbb{P}_{v_1,\ldots,v_i,\ldots,v_n}$.

[6] By leveraging hybrid encryption, we can transform any PE into one with $\mathsf{poly}(\lambda) + |m|$ ciphertext expansion, i.e., $\mathsf{Enc}'(\mathsf{mpk}, x, m) = \mathsf{Enc}(\mathsf{mpk}, x, s) || \mathsf{PRG}(s) \oplus m$ where $s \leftarrow\!\!\$\ \{0,1\}^\lambda$.

[7] When we write CPA secure PE, without specifying 1-sided or 2-sided security, we refer to a PE scheme that guarantees only the secrecy of the message. CPA secure PE is the same as CPA secure ABE.

input chosen by $\mathsf{A}$ during the challenge phase, and $b$ is the challenge bit. Since $\mathbb{P}_{v_1,\ldots,v_n}(x_1^b,\ldots,x_n^b)$ is a conjunction of arbitrary predicates (see Eq. (1)), this implies that there exists an $i \in [n]$ such that $\mathbb{P}_{v_i}(x_i^b) = 0$ for every $i$-th decryption key $\mathsf{dk}_{v_i}$ obtained by $\mathsf{A}$. We can leverage this observation together with the CPA security of $\mathsf{PE}_i$ to do a first hybrid in which the challenger computes the $i$-th layer of the challenge ciphertext as $\mathsf{Enc}_i(\mathsf{mpk}_i, x_i^b, 0\ldots0)$. Now, since the lock $y$ is not encrypted anymore, we can use the security of lockable obfuscation to do a second hybrid in which the challenge ciphertext $\widetilde{\mathbb{C}}$ is simulated by using the simulator of lockable obfuscation. In this last hybrid, the challenge ciphertext does not depend on the bit $b$ sampled by the challenger.

Despite we focused the discussion on CPA-1-sided security, we stress that the same construction achieves CPA-2-sided security if the underlying $n$ single-input PE schemes $\mathsf{PE}_1,\ldots,\mathsf{PE}_n$ are CPA-2-sided secure, i.e., $\mathsf{Enc}(\mathsf{mpk}, x^0, m^0)$ and $\mathsf{Enc}(\mathsf{mpk}, x^1, m^1)$ are indistinguishable even when $\mathbb{P}_{v_1,\ldots,v_n}(x^0) = \mathbb{P}_{v_1,\ldots,v_n}(x^1) = 1$ and $m^0 = m^1$.

*Multi-input Predicate Encryption.* We now turn to the more challenging setting of multi-input PE.[8] Here, each of the $n$ senders can use its corresponding encryption key to independently encrypt messages under different inputs for the predicate. For this reason, the setup algorithm of $n$-input PE outputs $n$ encryption keys $(\mathsf{ek}_1,\ldots,\mathsf{ek}_n)$ and a master secret key $\mathsf{msk}$. Each encryption key $\mathsf{ek}_i$ is given to the $i$-th sender and allows the latter to handle the $i$-th slot of a multi-input predicate. The $i$-th party encrypts a message $m_i$ under an input $x_i$ by using its encryption key $\mathsf{ek}_i$, i.e., $c_i \leftarrow_{\$} \mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$. On the other hand, a receiver can use the decryption key $\mathsf{dk}_{\mathbb{P}}$ associated to an $n$-input predicate $\mathbb{P}$ (recall that $\mathsf{dk}_{\mathbb{P}}$ is generated by the authority via $\mathsf{KGen}(\mathsf{msk}, \mathbb{P})$) to execute $\mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, c_1,\ldots,c_n)$. Intuitively, the decryption algorithm returns $(m_1,\ldots,m_n)$ when $\mathbb{P}(x_1,\ldots,x_n) = 1$ where $(m_i, x_i)$ are the message and the input associated to the $i$-th ciphertext $c_i$.

The CPA-1-sided security of $n$-input PE is similar to that of $n$-key PE, but adapted to the multi-input setting. Informally, an adversary $\mathsf{A}$ must not be able to distinguish between ciphertexts $(\mathsf{Enc}(\mathsf{ek}_i, x_i^0, m_i^0))_{i\in[n]}$ and $(\mathsf{Enc}(\mathsf{ek}_i, x_i^1, m_i^1))_{i\in[n]}$ where $(x_1^0,\ldots,x_n^0)$, $(x_1^1,\ldots,x_n^1)$ and $(m_1^0,\ldots,m_n^0)$, $(m_1^1,\ldots,m_n^1)$ are chosen by $\mathsf{A}$. Naturally, this is subject to the usual validity condition, informally saying that $\mathsf{A}$ should not be able to decrypt (part of) the challenge ciphertext. This condition can assume different meanings depending on whether the encryption keys are all secret or some of them are public (or can be leaked). Because of this, we formalize security with and without corruptions. Throughout the rest of this section, we describe how CPA-1-sided security of $n$-input PE changes in these two settings, and give some intuition on our constructions for each setting.

*Security in the Secret-Key Setting.* Here, no corruptions are allowed and thus the encryption keys are all secrets. Hence, an adversary $\mathsf{A}$ playing the CPA-1-sided security game has adaptive oracle access to both the key generation

[8] Indeed, as we discuss in Remark 1, CPA-1-sided (resp. CPA-2-sided) secure multi-input PE for arbitrary predicates implies CPA-1-sided (resp. CPA-2-sided) secure multi-key PE.

oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$ and to $n$ encryption oracles $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i\in[n]}$. The latter oracles allow $\mathsf{A}$ to generate ciphertexts (associated to the $i$-th input/sender) on adversarially chosen predicate inputs and messages. Since these ciphertexts are created independently, the adversary has the power to interleave part of the challenge ciphertext $(c_1^*, \dots, c_n^*)$ with the ciphertexts obtained trough the encryption oracles. This has a huge impact on the security of the a $n$-input PE scheme and on the validity condition that $\mathsf{A}$ must satisfy. For example, during the challenge phase, $\mathsf{A}$ could choose two vectors of messages $(m_1^0, \dots, m_n^0)$ and $(m_1^1, \dots, m_n^1)$ and two vectors of predicate inputs $(x_1^0, \dots, x_n^0)$ and $(x_1^1, \dots, x_n^1)$ such that for every predicate $\mathbb{P}$ (submitted to oracle $\mathsf{KGen}(m, \cdot)$) we have $\mathbb{P}(x_1^0, \dots, x_n^0) = \mathbb{P}(x_1^1, \dots, x_n^1) = 0$. Although the vector $(c_1^*, \dots, c_n^*)$ can not be directly decrypted, $\mathsf{A}$ could still be able to decrypt part of it by leveraging the encryption oracles. In more details, $\mathsf{A}$ could: (i) adversarially choose $x_i'$ such that $\mathbb{P}(x_1^0, \dots, x_i', \dots x_n^0) = 1$ and $\mathbb{P}(x_1^1, \dots, x_i', \dots x_n^1) = 0$; (ii) submit $(x_i', m_i')$ to oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$ and obtain $c_i'$;and (iii) simply decrypt the vector $(c_1^*, \dots, c_i', \dots, c_n^*)$. When $b = 0$ (resp. $b = 1$), the adversary knows that the challenge ciphertext must (resp. must not) decrypt successfully. This allows it to easily win the CPA-1-sided security experiment of $n$-input PE. As a consequence, the condition defining when $\mathsf{A}$ is valid depends on both the queries submitted to $\mathsf{KGen}(\mathsf{msk}, \cdot)$ and to the oracles $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i\in[n]}$. More precisely, for every decryption key $\mathsf{dk}_{\mathbb{P}}$ corresponding to a predicate $\mathbb{P}$, for every vector of ciphertexts obtained by interleaving the challenge ciphertext $(c_1^*, \dots, c_n^*)$ with the ciphertexts generated trough any of the $n$ encryption oracles, we must have that $\mathbb{P}$ is not satisfied. This is formalized by the following condition: $\forall \mathbb{P} \in \mathcal{Q}_{\mathsf{KGen}}$, $\forall j \in [n]$, $\forall i_1 \in [k_1 + 1], \dots, \forall i_n \in [k_n + 1]$, it holds that

$$\begin{aligned}&\mathbb{P}(x_1^{(i_1,0)}, \dots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \dots, x_n^{(i_n,0)}) = \\ &\mathbb{P}(x_1^{(i_1,1)}, \dots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \dots, x_n^{(i_n,1)}) = 0, \qquad (2)\end{aligned}$$

where $\mathcal{Q}_{\mathsf{KGen}}$ are the queries submitted to oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$, $(x_1^0, \dots, x_n^0), (x_1^1, \dots, x_n^1)$ are the predicate inputs chosen by $\mathsf{A}$ during the challenge phase, and $\mathcal{Q}_i^b = \{x_i^{(1,b)}, \dots, x_i^{(k_i,b)}, x_i^{(k_i+1,b)} = x_i^b\}$ is the ordered list composed of the $k_i$ predicate inputs submitted to oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$ and the challenge input $x_i^b$ for $b \in \{0, 1\}, i \in [n]$ (observe that $\mathcal{Q}_i^0$ and $\mathcal{Q}_i^1$ are identical except for the last element). The formal security definition appears in Sect. 4.

*Construction in the Secret-Key Setting.* We propose a construction of $n$-input PE for conjunctions of arbitrary predicates (see Eq. (1)) *with wildcards* from single-input PE, lockable obfuscation, and SKE. In particular, we start from single-input PE for arbitrary predicates. Actually, it will suffice that the underlying PE itself supports the predicates $\mathbb{P}(x_1, \dots, x_n)$ as defined in Eq. (1), where we view $(x_1, \dots, x_n)$ as a single input chosen by the sender. In addition, the predicate must have a (efficiently computable) wildcard input $(x_1^\star, \dots, x_n^\star)$ such that $x_i^\star$ satisfies every $i$-th predicate of the conjunction, i.e., $\mathbb{P}_i(x_i^\star) = 1$. As we will describe next, the $n-1$ subset of wildcards $(x_1^\star, \dots, x_{i-1}^\star, x_{i+1}^\star, \dots, x_n^\star)$ will permit the $i$-th sender to put a "don't care" placeholder on the slots of the other senders. This will allow the construction to deal with multiple inputs without compromising the evaluation of the predicate.

The main intuition behind our construction is to evaluate the conjunction of the predicates inside lockable obfuscation in such a way that, as soon as one of the predicates (of the conjunction) is not satisfied, both the messages and the predicate inputs remain hidden (even if another predicate $\mathbb{P}_i$ is satisfied). To accomplish that, we need to create a link between the independently generated ciphertexts (each produced by different senders). This is done by leveraging an SKE scheme as follows.

In a nutshell, the $i$-th secret encryption key has the form $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$ where $\mathsf{mpk}$ is the master public key of the single-input PE, and $\mathsf{k}_i$ for $i \in [n]$ is a secret key for the SKE (we also let $\mathsf{ek}_{n+1} = \mathsf{k}_1$). In order to encrypt a message $m_i$ under an input $x_i$, the $i$-th sender samples a random lock $y_i$ and encrypts $(y_i, \mathsf{k}_{i+1})$ via the single-input PE, using the input made by all the wildcards $x_j^\star$ except for the position $j = i$, where, instead, the sender places its real input $x_i$, i.e., $c_i^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, (x_1^\star, \ldots, x_{i-1}^\star, x_i, x_{i+1}^\star, \ldots, x_n^\star), (y_i, \mathsf{k}_{i+1}))$. The final ciphertext $c_i$ will be $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$, where $c_i^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{k}_i, c_i^{(1)})$ and $\widetilde{\mathbb{C}}_i$ is the obfuscation of the circuit $\mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}$ under the lock $y_i$ and message $m_i$. Similarly to the case of multi-key PE, the latter circuit is responsible for the decryption. In particular, upon input the ciphertexts $(c_{i+1}^{(2)}, \ldots, c_n^{(2)}, c_1^{(2)}, \ldots, c_{i-1}^{(2)})$—note the order of the ciphertexts—and the decryption key $\mathsf{dk}_{\mathbb{P}}$ for $\mathbb{P}(x_1, \ldots, x_n)$, the circuit $\mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}$ acts as follows:

1. Set $\mathsf{k} = \mathsf{k}_{i+1}$ where $\mathsf{k}_{i+1}$ is the secret key hardcoded into the circuit.
2. For $c_j^{(2)} \in \{c_{i+1}^{(2)}, \ldots, c_n^{(2)}, c_1^{(2)}, \ldots, c_{i-1}^{(2)}\}$ do:
   (a) Decrypt $c_j^{(2)}$ using the secret key $\mathsf{k}$, i.e., $c_j^{(1)} = \mathsf{Dec}(\mathsf{k}, c_j^{(2)})$.
   (b) Decrypt $c_j^{(1)}$ using $\mathsf{dk}_{\mathbb{P}}$ in order to get $(y_j, \mathsf{k}_{j+1})$. If $c_j^{(1)}$ decrypts correctly, $\mathsf{k}_{j+1}$ is the secret key used to encrypt the next ciphertext $c_{j+1}^{(2)}$.
   (c) Set $\mathsf{k} = \mathsf{k}_{j+1}$.
3. Compute $(y_i, \mathsf{k}_{i+1}) = \mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, \mathsf{Dec}(\mathsf{k}, c_i^{(2)}))$, where $c_i^{(2)}$ is the ciphertext hardcoded into the circuit.
4. Return $y_i$ (note that if none of the decryptions fails then $y_i$ is the lock used to obfuscate the circuit).

By the above description, decryption is immediate: Upon input $(c_i)_{i\in[n]}$, the receiver computes $m_i = \widetilde{\mathbb{C}}_i(c_{i+1}^{(2)}, \ldots, c_n^{(2)}, c_1^{(2)}, \ldots, c_{i-1}^{(2)}, \mathsf{dk}_{\mathbb{P}})$ where $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$ and $\mathsf{dk}_{\mathbb{P}}$ is the decryption key of the underlying single-input PE for a predicate $\mathbb{P}(x_1, \ldots, x_n)$. We highlight that the combination of the SKE with the PE wildcards is what allows our construction to correctly implement the predicates of Eq. (1). This is because, when $c_i^{(1)}$ correctly decrypts under the key $\mathsf{dk}_{\mathbb{P}}$ (Item 2b), we are guaranteed that $\mathbb{P}_i(x_i) = 1$ (recall that $x_i$ is the input of the $i$-th sender). In particular, the latter holds as, in any other slot, the $i$-th sender has used the wildcards. By repeating this argument, we can conclude that $\mathbb{P}(x_1, \ldots, x_n) = \mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)$ is satisfied if the execution of each $\mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}$ goes as expected. We refer the reader to the full version [25] for the formal construction.

As for security, we show that our construction satisfies CPA-1-sided security in the presence of *no collusions* (i.e., the adversary can submit a single query to the oracle $\mathsf{KGen}$) if the underlying PE is CPA-1-sided secure, SKE is CPA secure, and the lockable obfuscation is secure. Roughly, the proof works as follows. Let $\mathbb{P}^*$ be the *only* predicate submitted to $\mathsf{KGen}$ by the adversary. Starting from $\mathsf{A}$'s validity condition, we infer that, for any choice of the challenge bit $b \in \{0, 1\}$, then attacker $\mathsf{A}$ must maintain one of the following two conditions:

(i) either $\mathbb{P}_1^*(x_1^b) = \ldots = \mathbb{P}_n^*(x_n^b) = 0$ (i.e., all the predicates of the conjunctions are false);
(ii) or (if at least one predicate $\mathbb{P}_i^*$ is satisfied, i.e., $\mathbb{P}_i^*(x_i^b) = 1$) there exists $j \neq i$ such that, for every $x_j \in \mathcal{Q}_j^b$, it holds that $\mathbb{P}_j^*(x_j) = 0$ where $\mathcal{Q}_j^b$ is the ordered list composed of predicate inputs submitted to the oracle $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ and the challenge input $x_j^b$ (see Eq. (2)).[9]

When the first condition is satisfied, we can leverage the CPA-1-sided security of the single-input PE to show that the every lock $y_i$ (encrypted using the PE), and every input $x_i$ (encrypted in $c_i^{(2)}$), is completely hidden to the adversary. The latter allows us to use the security of lockable obfuscation to move to a hybrid experiment in which all the (obfuscated) circuits are simulated (including the messages).

On the other hand, when the second condition is satisfied, we can transition to a hybrid experiment (this time by leveraging the security of the underlying PE scheme) in which $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ computes $c_j^{(1)}$ by encrypting the all-zero string (instead of $(y_j, \mathsf{k}_{j+1})$). Thus, we can use the security of lockable obfuscation to move to another hybrid in which $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ simulates all the obfuscations. At this point, the symmetric key $\mathsf{k}_{j+1}$ is not used anymore. Hence, we can use the security of SKE to transition to another hybrid in which $\mathsf{Enc}(\mathsf{ek}_{j+1}, \cdot, \cdot)$ computes $c_{j+1}^{(2)}$ by encrypting the all-zero string (instead of $c_{j+1}^{(1)}$ that, in turn, contains the lock $y_{j+1}$ and the symmetric key $\mathsf{k}_{j+2}$). After this hybrid, we can again use the security of lockable obfuscation to simulate all the obfuscations computed by $\mathsf{Enc}(\mathsf{ek}_{j+1}, \cdot, \cdot)$, and so on. By repeating these last two hybrids, we reach an experiment whose distribution does not depend on the challenge bit. The formal construction appears in the full version of this work [25].

We highlight that our scheme is not secure in the presence of collusions. In particular, the fact that the adversary can obtain a single decryption key $\mathsf{dk}_{\mathbb{P}}$ is crucial in order to get the validity condition (ii), i.e., for every $b \in \{0, 1\}$ there exists a $j$ such that for every predicate (submitted to $\mathsf{KGen}(\mathsf{msk}, \cdot)$) we have $\mathbb{P}_j(x_j^b) = 0$. In fact, in the case of collusions, the adversary can ask for two decryption keys $\mathsf{dk}_{\mathbb{P}}$ and $\mathsf{dk}_{\mathbb{P}'}$ such that for every $b \in \{0, 1\}$:

[9] If this condition is not satisfied, the adversary has obtained through the encryption oracles a set of ciphertexts that can be interleaved with one (or more) parts of the challenge ciphertext in order to satisfy the predicate $\mathbb{P}^*$.

$$\mathbb{P}_1(x_1^b) = 0 \text{ and } \mathbb{P}_2(x_2^b) = \ldots = \mathbb{P}_n(x_n^b) = 1$$
$$\mathbb{P}'_1(x_1^b) = 1 \text{ and } \mathbb{P}'_2(x_2^b) = \ldots = \mathbb{P}'_n(x_n^b) = 0.$$

Note that these are valid queries for the CPA-1-sided security experiment of $n$-input PE (the ciphertext cannot be decrypted). However, such a *unique* $j$ for every predicate (as per condition (ii)) does not exist. When this happens, we are not able to conclude the proof by making a reduction to the security of single-input PE (the reduction will make an invalid set of queries to the $\mathsf{KGen}$ oracle of the single-input PE, making it invalid for the CPA-1-sided security of the single-input PE).[10]

Lastly, we stress that since we start from a single-input PE supporting conjunctions of arbitrary predicates *with wildcards*, we end up with an $n$-input PE for conjunctions of arbitrary predicates (see Eq. (1)) *with wildcards*. We highlight that wildcards do not play any role in the security proof of our secret-key construction. In other words, wildcards are required for functionality (correctness) and not for security. Indeed, in the secret-key setting (i.e., no corruptions), wildcards can be easily removed. This is because we can transform any secure multi-input PE for $\mathbb{P}(x_1, \ldots, x_n) = \mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)$ with a *single* wildcard $(x_1^\star, \ldots, x_n^\star)$ into a secure multi-input PE for the same class of predicates $\mathbb{P}(x_1, \ldots, x_n)$ *without the wildcard*. This can be done by requiring the senders not to encrypt the corresponding wildcard, i.e., for each $i \in [n]$, $\mathsf{Enc}(\mathsf{ek}_i, x_i^\star, m_i)$ outputs $\perp$ whenever $x_i = x_i^\star$. We stress that this only works in the case of no corruptions. In fact, as we will discuss later, in case of corruption, wildcards play a role in the security of our corruption-resilient multi-input PE scheme, e.g., an adversary can encrypt wildcards on its own using the leaked encryption keys.

*Security Under Corruptions.* Next, let us explain how to define security of multi-input PE in the presence of corruptions. Here, the adversary has the possibility to corrupt a subset of the senders and leak their encryption keys $\mathsf{ek}_i$. We model this by introducing an additional corruption oracle $\mathsf{Corr}(\cdot)$ that, upon input an index $i \in [n]$, returns $\mathsf{ek}_i$. Note that, once obtained $\mathsf{ek}_i$, the adversary $\mathsf{A}$ has the possibility to produce arbitrary ciphertexts on any message and predicate input, without interacting with the challenger during the CPA-1-sided security game. As usual, the validity condition heavily depends on the queries submitted to both the encryption oracles and the corruption oracle. More precisely, the validity condition now says that, for every decryption key $\mathsf{dk}_\mathbb{P}$, for every vector of ciphertexts that can be obtained by interleaving the challenge ciphertext $(c_1^*, \ldots, c_n^*)$ with both the ciphertexts obtain trough any of the (uncorrupted) encryption oracles and the ones that $\mathsf{A}$ may autonomously produce by using the leaked encryption keys (trough oracle $\mathsf{Corr}(\cdot)$), we have that $\mathbb{P}$ is not satisfied. Hence, the validity condition is identical to that of the secret-key setting (see Eq. (2)), except that:

[10] As we discuss in the full version [25], our construction remains secure if we consider a weaker form of collusion in which the adversary can only obtain multiple decryption keys for predicates $\mathbb{P}$ such that there is a unique $j$ *for all predicates* (submitted to $\mathsf{KGen}$) that satisfies the validity condition (ii).

– If the $i$-th encryption key $\mathsf{ek}_i$ has been corrupted/leaked, then $\mathcal{Q}_i^b$ of Eq. 2 corresponds to the $i$-th predicate input space. This is because the adversary can produce a valid ciphertext on any input $x_i$.
– Else (i.e., the $i$-th encryption key $\mathsf{ek}_i$ is still secret), $\mathcal{Q}_i^b$ is defined as usual, i.e., it is the ordered list of predicate inputs submitted to oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$ and challenge input $x_i^b$.

See Sect. 4 for the formal definition.

*A Simple Attack.* Before explaining our construction in details, let us show why the previous construction is not secure under corruptions. For simplicity, we focus on the 2-input setting. Suppose an adversary $\mathsf{A}$ has a single decryption key $\mathsf{dk}_{\mathbb{P}}$ for $\mathbb{P}(x_1, x_2) = \mathbb{P}_1(x_1) \wedge \mathbb{P}_2(x_2)$ and a vector of ciphertexts $(c_1^*, c_2^*) = ((\widetilde{\mathbb{C}}_1, c_1^{(2)}), (\widetilde{\mathbb{C}}_2, c_2^{(2)}))$ encrypted under the predicate input $(x_1, x_2)$ such that $\mathbb{P}_1(x_1) = 0$ and $\mathbb{P}_2(x_2) = 1$. Note that this ciphertext should not decrypt under $\mathsf{dk}_{\mathbb{P}}$, since the conjunction of $\mathbb{P}_1$ and $\mathbb{P}_2$ evaluates to 0. If $\mathsf{A}$ can obtain $\mathsf{ek}_2$, then it can easily determine the message $m_2$ (and thus the bit $b$). Indeed, once $\mathsf{A}$ gets $\mathsf{ek}_2 = (\mathsf{mpk}, \mathsf{k}_2, \mathsf{k}_1)$, it can compute a malicious ciphertext $\widetilde{c}_1^{(1)}$ (using the single-input PE) by encrypting $(\widetilde{y}, \mathsf{k}_2)$ (where $\widetilde{y}$ is a random lock) under the predicate input composed by $(x_1', x_2')$ such that $\mathbb{P}_1(x_1') = 1$ and $\mathbb{P}_2(x_2') = 1$. Then, it can compute $\widetilde{c}_1^{(2)} \leftarrow\$ \mathsf{Enc}(\mathsf{k}_1, \widetilde{c}_1^{(1)})$ and execute $\widetilde{\mathbb{C}}_2(\widetilde{c}_1^{(2)}, \mathsf{dk}_{\mathbb{P}})$ to get $m_2$. Note that by definition the execution of $\widetilde{\mathbb{C}}_2$ outputs the correct message, since $\mathbb{P}_1(x_1^\star) \wedge \mathbb{P}_2(x_2) = 1$ and $\widetilde{c}_1^{(2)}$ contains the correct secret encryption key $\mathsf{k}_2$, allowing the circuit to correctly end the computation. Also, note that this attack does not violate the validity condition. This is because $\mathbb{P}_1(x_1) = 0$, and $\mathsf{A}$ does not use the oracle $\mathsf{Enc}(\mathsf{ek}_1, \cdot, \cdot)$ at all. Hence, any interleaving of the ciphertexts will involve the predicate input $x_1$ that, in turn, will make the conjunction $\mathbb{P}(x_1, x_2') = \mathbb{P}_1(x_1) \wedge \mathbb{P}_2(x_2')$ unsatisfied for every choice of the input predicate $x_2'$.

In light of the above attack, we can identify what we need to do in order to extend our techniques to handle corruptions: First, following the proof of the previous construction, it is important to hide the (plain) single-input PE ciphertext that a particular sender produces (e.g., in the secret-key setting we re-encrypt $c_i^{(1)}$ using SKE). As we have described for the secret-key setting, this allows us to claim that everything remains hidden whenever one of the predicate $\mathbb{P}_i$ of the conjunction is not satisfied (even if a different $\mathbb{P}_j$ is satisfied).[11] Second, the leakage of one (or more) encryption keys should not allow to produce a malicious ciphertext on behalf of the uncorrupted senders (or simply decrypt the ciphertexts of other parties). Otherwise, the attacker can follow a strategy similar to the one above to break security.

[11] The secret-key construction achieves this by linking multiple PE ciphertexts via SKE, and including the secret key $\mathsf{k}_{i+1}$ into the PE ciphertext.

*Construction Under Corruptions.* In order to achieve the above properties, we propose a new technique based on nested (lockable obfuscated) circuits that can be executed one inside the other. This technique permits to make available secret information (e.g., secret keys) *only* during nested execution. For the sake of clarity, we first present our approach for the case of two inputs. As an initial attempt to deal with corruptions, we replace the SKE in our previous construction with a PKE, so that the encryption key $\mathsf{ek}_1$ (resp. $\mathsf{ek}_2$) is now composed of $(\mathsf{mpk}, \mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2)$ (resp. $(\mathsf{mpk}, \mathsf{sk}_2, \mathsf{pk}_2, \mathsf{pk}_1)$) where $(\mathsf{sk}_i, \mathsf{pk}_i)$ is a secret/public key pair. Each $(\mathsf{sk}_i, \mathsf{pk}_i)$ is associated to the $i$-th sender (indeed, note that $\mathsf{ek}_i$ contains also the secret key $\mathsf{sk}_i$). From the perspective of the first sender, in order to encrypt a message $m_1$ under the input $x_1$, it samples two random locks $(y_1^{\mathsf{in}}, y_1^{\mathsf{out}})$ and encrypts them (using the single-input PE) as before using the wildcard $x_2^\star$, i.e., $c_1^{(0)} \leftarrow\$ \mathsf{Enc}(\mathsf{mpk}, (x_1, x_2^\star), (y_1^{\mathsf{in}}, y_1^{\mathsf{out}}))$.[12] At this point, the PE ciphertext $c_1^{(0)}$ is re-encrypted twice using $\mathsf{pk}_1$ and $\mathsf{pk}_2$, i.e., $c_1^{(i)} \leftarrow\$ \mathsf{Enc}(\mathsf{pk}_i, c_1^{(i-1)})$ for $i \in [2]$. Intuitively, the two layers of PKE have the role of hiding the PE ciphertexts (that in turn contain the locks) even when the adversary leaks all encryption keys except one. The final ciphertext is composed by the two obfuscations $\widetilde{\mathbb{C}}_1^{\mathsf{out}}$, $\widetilde{\mathbb{C}}_1^{\mathsf{in}}$ of the circuits $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(2)}}$, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(2)}}$, respectively. The former is obfuscated under the lock $y_1^{\mathsf{out}}$ and message $m_1$, whereas the latter is obfuscated under the lock $y_1^{\mathsf{in}}$ and message $\mathsf{sk}_1$. The ciphertext produced by the second sender, is identical, except that it uses $\mathsf{sk}_2$ (instead of $\mathsf{sk}_1$) and that $c_2^{(0)}$ is computed using the predicate input $(x_1^\star, x_2)$ (instead of $(x_1, x_2^\star)$).

The crux of our nesting technique comes from the definition of the circuits $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_i, c_i^{(2)}}$. More precisely, the outer circuit $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(2)}}$ will take as input the obfuscation $\widetilde{\mathbb{C}}_2^{\mathsf{in}}$ of the inner circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(2)}}$ and a decryption key $\mathsf{dk}_{\mathbb{P}}$. Then, in order to securely check the conjunction inside the lockable obfuscation, $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(2)}}$ will execute $\widetilde{\mathbb{C}}_2^{\mathsf{in}}(\mathsf{sk}_1, \mathsf{dk}_{\mathbb{P}})$. At this point, $\widetilde{\mathbb{C}}_2^{\mathsf{in}}$ has everything it needs to check the satisfiability of $\mathbb{P}_2(\cdot)$. It removes the PKE layers from $c_2^{(2)}$ by computing $c_2^{(0)} = \mathsf{Dec}(\mathsf{sk}_2, \mathsf{Dec}(\mathsf{sk}_1, c_2^{(2)}))$. Then, it decrypts the PE ciphertext $(y_2^{\mathsf{in}}, y_2^{\mathsf{out}}) = \mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, c_2^{(0)})$—observe that the decryption succeeds if $\mathbb{P}_2(x_2) = 1$—and returns $y_2^{\mathsf{in}}$. By correctness of lockable obfuscation, if the computation of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(2)}}(\mathsf{sk}_1, \mathsf{dk}_{\mathbb{P}})$ goes as intended, then $\widetilde{\mathbb{C}}_2^{\mathsf{in}}(\mathsf{sk}_1, \mathsf{dk}_{\mathbb{P}})$ will output $\mathsf{sk}_2$ (the message attached to the obfuscation). Once obtained $\mathsf{sk}_2$, the computation of $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(2)}}$ can continue and perform a similar computation to check the satisfiability of $\mathbb{P}_1(\cdot)$ except that, if the PE ciphertext $c_1^{(0)}$ decrypts correctly, it returns $y_1^{\mathsf{out}}$. If all the decryptions (performed by $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(2)}}$ and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(2)}}$) succeed, the execution of the obfuscation $\widetilde{\mathbb{C}}_1^{\mathsf{out}}$ of $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(2)}}$ will output $m_1$. A symmetrical argument holds for $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_2, c_2^{(2)}}$ and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(2)}}$, releasing $m_2$.

[12] Recall that wildcards must be efficiently computable.

We show that the above 2-input PE construction is CPA-1-sided secure under 1 corruption (i.e., one encryption key remains secret) and no collusions if the underlying single-input PE is CPA secure, PKE is CPA secure, and the lockable obfuscation is secure. The high level intuition is that $\mathsf{sk}_i$ remains unknown to the adversary if $\mathbb{P}_i(\cdot) = 0$ (unless the adversary invokes the oracle $\mathsf{Corr}(i)$). This is reflected by the proof technique that is sketched below.

Let $\mathsf{dk}_{\mathbb{P}^*}$ be the decryption key obtained by $\mathsf{A}$ for the predicate $\mathbb{P}^*(\cdot,\cdot) = \mathbb{P}^*_1(\cdot) \wedge \mathbb{P}^*_2(\cdot)$ (recall the presence of wildcards), and let $\mathcal{Q}_{\mathsf{Corr}}$ be the queries submitted to the corruption oracle. Starting from the validity condition, we can infer that for any choice of the challenge bit $b \in \{0,1\}$ we have:

(i) either $\mathbb{P}^*_1(x^b_1) = \mathbb{P}^*_2(x^b_2) = 0$;
(ii) or (i.e., there exists an $i \in [2]$ such that predicate $\mathbb{P}_i$ is satisfied) $j \notin \mathcal{Q}_{\mathsf{Corr}}$ such that $j \neq i$ and, for every $x_j \in \mathcal{Q}^b_j$, $\mathbb{P}^*_j(x_j) = 0$ (recall that $x^b_j \in \mathcal{Q}^b_j$). Observe that this second condition holds because of the following:
  - If there is $x_j \in \mathcal{Q}^b_j$ such that $\mathbb{P}^*_j(x_j) = 1$, $\mathsf{A}$ can use the corresponding ciphertext to decrypt the $i$-th part of the challenge ciphertext since $\mathbb{P}^*_i(x^b_i) = 1$.
  - If $j \in \mathcal{Q}_{\mathsf{Corr}}$, $\mathsf{A}$ can simply use $\mathsf{ek}_j$ to encrypt a random message under the wildcard $x^\star_j$ (that always exists by design of our construction) and, again, decrypt the $i$-th part of the challenge ciphertext. Note that, contrarily from our secret-key construction, wildcards play an important role in the security of our multi-input PE construction under corruptions (if an encryption key $\mathsf{ek}_j$ gets leaked then a malicious adversary can always encrypt itself the $j$-th wildcards $x^\star_j$, satisfying the $j$-th predicate $\mathbb{P}_j$). Hence, in the corruption setting, wildcards are used for both functionality and security.

By leveraging the above two conditions, the security of our scheme follows by using a similar argument to that of the secret-key setting. In particular, when the first condition is satisfied, we can show that the locks $(y^{\mathsf{in}}_1, y^{\mathsf{out}}_1)$ and $(y^{\mathsf{in}}_2, y^{\mathsf{out}}_2)$ (used to encrypt the challenge) are completely hidden. This, in turn, allows us to use the security of lockable obfuscation and simulate the obfuscations of $(\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1,c^{(2)}_1}, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1,c^{(2)}_1})$, $(\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_2,c^{(2)}_2}, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2,c^{(2)}_2})$, and the corresponding messages.

On the other hand, when the second condition is satisfied, we can move to a hybrid (by leveraging the security of single-input PE) in which $\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)$ computes $c^{(0)}_j$ by encrypting the all-zero string (instead of $(y^{\mathsf{in}}_j, y^{\mathsf{out}}_j)$). Then, we can use the security of lockable obfuscation to transition to another hybrid in which $\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)$ simulates all the obfuscations. At this point, the secret key $\mathsf{sk}_j$ of the uncorrupted $j$-th sender is not used anymore (recall that $j \notin \mathcal{Q}_{\mathsf{Corr}}$). Hence, we can leverage the security of the PKE to remove the locks $(y^{\mathsf{in}}_i, y^{\mathsf{out}}_i)$ chosen by the $i$-th sender (recall $i \neq j$). In more details, we do another hybrid in which the $j$-th PKE layer $c^{(j)}_i$ of the challenge ciphertext is an encryption of zeroes (instead of $c^{(j-1)}_i$ that, in turn, encrypts the locks $(y^{\mathsf{in}}_i, y^{\mathsf{out}}_i)$). After this hybrid, we can again use the security of lockable obfuscation to simulate all the obfuscations (and the corresponding attached messages) that compose the $i$-th component of the ciphertext. The distribution of this last hybrid does not depend on the challenge bit $b$ since all the ciphertexts are simulated by the simulator of the lockable obfuscation scheme.

To sum up, we can observe that encrypting $c_i^{(0)}$ (the PE ciphertext that contains the locks) with the public keys $(\mathsf{pk}_1, \mathsf{pk}_2)$ of both senders is crucial in order for our proof to work independently of which encryption key the adversary decides to leak. So long as at least one encryption key $\mathsf{ek}_i$ remains hidden, then there is a PKE layer that cannot be decrypted by the adversary. This allows the proof to go through.

*Generalizing the Nesting Technique to* $(n > 2)$ *Inputs.* By carefully modifying the definition of the outer and inner circuits, we can generalize the above technique to the case of $n > 2$. The structure of the encryption keys and of the encryption algorithm is similar to the case $n = 2$:

- Each encryption key $\mathsf{ek}_i$ is of the form $(\mathsf{mpk}, \mathsf{sk}_i, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$.
- To compute the $i$-th encryption of $(x_i, m_i)$, the sender computes the initial PE ciphertext as $c_i^{(0)} \leftarrow\$ \mathsf{Enc}(\mathsf{mpk}, (x_1^\star, \ldots, x_i, \ldots, x_n^\star), (y_i^{\mathsf{in}}, y_i^{\mathsf{out}}))$. Then, it re-encrypts $n$ times the ciphertext $c_i^{(0)}$ using $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, i.e., $c_i^{(v)} \leftarrow\$ \mathsf{Enc}(\mathsf{pk}_v, c_i^{(v-1)})$ for $v \in [n]$. As usual, the final ciphertext $c_i = (\widetilde{\mathbb{C}}_i^{\mathsf{out}}, \widetilde{\mathbb{C}}_i^{\mathsf{in}})$ is composed of the obfuscations of $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_i, c_i^{(n)}}$ and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_i, c_i^{(n)}}$.

We now turn on the crucial point: the definition of the outer and inner circuits. Again, for the sake of clarity, we only describe the outer circuit $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(n)}}$ and of the inner circuits $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(n)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n, c_n^{(n)}})$ generated by the corresponding senders. The remaining circuits are defined similarly. First off, the input space of these circuits is a follows:

- $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(n)}}$ takes as input the $n-1$ obfuscations of the circuits $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(n)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n, c_n^{(n)}})$ and a decryption $\mathsf{dk}_{\mathbb{P}}$. These obfuscations are the inner circuits that needs to be executed in order to return the message $m_1$ attached to the obfuscation of $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(n)}}$.
- On the other hand, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_i, c_i^{(n)}}$, for $i \in [n] \setminus \{1\}$, takes as input a tuple of $n$ secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ (where some can be set to $\perp$), a decryption key $\mathsf{dk}_{\mathbb{P}}$, and the obfuscations of $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{i+1}, c_{i+1}^{(n)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n, c_n^{(n)}})$. Intuitively, these obfuscations are the remaining inner circuits that we need to still execute in order to complete the nested execution.

Intuitively, the decryption of $m_1$ requires the nested execution of these circuits (starting from the outer one) in order to get all the secret keys required to decrypt the PE ciphertext. This is achieved as follows.

The outer circuit $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1,c_1^{(n)}}$ starts the nested execution by invoking the obfuscation of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2,c_2^{(n)}}$ upon input $(\mathsf{sk}_1,\bot,\ldots,\bot)$, $\mathsf{dk}_{\mathbb{P}}$, and the remaining obfuscations of $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_3,c_3^{(n)}},\ldots,\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}})$. In turn, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2,c_2^{(n)}}$ will do a similar thing: It executes the next obfuscated circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_3,c_3^{(n)}}$ upon input $(\mathsf{sk}_1,\mathsf{sk}_2,\bot,\ldots,\bot)$, $\mathsf{dk}_{\mathbb{P}}$, and the remaining obfuscations $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_4,c_4^{(n)}},\ldots,\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}})$. This process is repeated until $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$ is executed upon input $(\mathsf{sk}_1,\ldots,\mathsf{sk}_{n-1},\bot)$ and $\mathsf{dk}_{\mathbb{P}}$. At this point, all the secret keys are know (observe that $\mathsf{sk}_n$ is hardcoded). From $c_n^{(n)}$, we can remove the $n$ PKE layers, decrypt the PE ciphertext and, in turn, return $y_n^{\mathsf{in}}$ if the PE ciphertext decrypts correctly (i.e., $\mathbb{P}_n(\cdot)$ is satisfied). Once $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$ terminates, the secret key $\mathsf{sk}_n$ is released and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_{n-1}^{(n)}}$ performs the computation required to check if $\mathbb{P}_{n-1}(\cdot)$ is satisfied. Indeed, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_{n-1}^{(n)}}$ has been executed on input $(\mathsf{sk}_1,\ldots,\mathsf{sk}_{n-2},\bot,\bot)$, it has $\mathsf{sk}_{n-1}$ harcoded, and the execution of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$ has released $\mathsf{sk}_n$. Hence, after the correct termination of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$, all secret keys are known.

It may seems that this argument can be iterated. However, there is a problem. Even if $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_{n-1}^{(n)}}$ correctly terminates, the circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-2},c_{n-2}^{(n)}}$ that invokes it does not have access to the secret key $\mathsf{sk}_n$. This is because the latter circuit receives as input $(\mathsf{sk}_1,\ldots,\mathsf{sk}_{n-3},\bot,\bot,\bot)$, it has $\mathsf{sk}_{n-2}$ hardcoded, and the circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_n^{(n)}}$ has returned $\mathsf{sk}_{n-1}$. As a consequence, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-2},c_{n-2}^{(n)}}$ must re-run $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$ on input $(\mathsf{sk}_1,\ldots,\mathsf{sk}_{n-1},\bot)$ in order to get $\mathsf{sk}_n$ and decrypt every PKE layer. This needs to be done at any level of the nested execution, yielding an asymptotic running time of $O(n^n)$. Hence, this technique only works assuming $n = O(1)$, i.e. for $O(1)$-input predicates. The formal construction is described in Sect. 5.2.

*On Achieving CPA-2-Sided Secure Multi-input PE.* Until now, we only focused the discussion on achieving CPA-1-sided security. Our multi-input constructions achieve CPA-2-sided security if the underlying single-input PE is CPA-2-sided secure (we highlight that, in our secret-key multi-input PE construction, we need to reduce the $n$-arity from $\mathsf{poly}(\lambda)$ to $O(\log(\lambda))$ since we use complexity leveraging). We just recall here that, already for the simple notion of single-input PE for arbitrary predicates, CPA-2-sided security implies iO [15].

## 1.3 Applications

Finally, we explore applications of multi-key and multi-input PE. This question is particularly relevant given the fact that we are only able to obtain multi-key and multi-input PE supporting conjunctions of arbitrary predicates (with wildcards). Luckily, we can show that this class of predicates is already expressive enough to yield interesting cryptographic applications which previously required much stronger assumptions. We refer the reader to the full version [25] for more details.

*Matchmaking Encryption (ME).* ME is a natural generalization of ABE in which both the sender and the receiver can specify their own attributes and access policies. Previous work showed how to obtain CPA-2-sided (i.e., mismatch and match) secure ME for arbitrary policies with unbounded collusions using iO [10,11], or for very restricted policies (i.e., for identity matching) using bilinear maps [20,26] (and ROM [10]). To this end, our CPA-1-sided secure multi-key PE scheme (from the sub-exponential LWE assumption) for conjunction of arbitrary predicates implies the weaker (and non-trivial) notion of CPA-1-sided secure ME (i.e., mismatch only). We stress that the seminal work of ME [10,11] defined ME in the setting of CPA-2-sided security (i.e., mismatch and match).

*Non-interactive MPC (NI-MPC).* NI-MPC [14,34] allows $n$ parties to evaluate a function $f(v_1, \ldots, v_n)$ on their inputs using a single round of communication (i.e., each party sends a single message $c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, v_i)$). This is achieved by assuming a trusted setup (that may depend on the function itself) that generates (possibly correlated) strings (e.g., common reference string $\mathsf{crs}$ and encryption keys $\mathsf{ek}_i$) that can be later used by the parties to perform function evaluation. Security is formalized using an indistinguishability-based definition: an adversary $\mathsf{A}$ cannot distinguish between $(\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, v_i^0))_{i\in[n]}$ and $(\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, v_i^1))_{i\in[n]}$, so long as any combination of the messages known by the adversary (including the ones it can compute using the encryption key $\mathsf{ek}_i$ of a corrupted party) yields the same function's evaluation.[13] Previous works [14,29,32,33] showed that NI-MPC implies iO even if we consider very weak security models, like the non-reusable 1-robust (i.e., one malicious party) setting with $n = 2$ parties, or the reusable 0-robust (i.e., no malicious parties) setting with $n = \mathsf{poly}(\lambda)$ parties.[14]

We show that CPA-1-sided multi-input PE supporting predicates $\mathbb{P}(x_1, \ldots, x_n)$ tolerating $k$ corruptions and no collusions implies $k$-robust NI-MPC for the following class of functions:

$$f_{\mathbb{P}}((x_1, m_1), \ldots, (x_n, m_n)) = \begin{cases} (m_1, \ldots, m_n) & \text{if } \mathbb{P}(x_1, \ldots, x_n) = 1 \\ \perp & \text{otherwise.} \end{cases}$$

The resulting NI-MPC satisfies a weaker notion of reusability without session identifiers (i.e., messages produced in different rounds can be interleaved by design) specifically tailored for all-or-nothing functions, which we name *CPA-1-sided reusability.* In a nutshell, CPA-1-sided reusable NI-MPC guarantees security even if the same setup is reused multiple times, so long as $f_{\mathbb{P}}$ outputs $\perp$ (i.e., $\mathbb{P}$ is not satisfied) for any combination of the honest messages and the ones the

[13] Note that security of NI-MPC for general functions is formalized by an indistinguishability-based definition [14,32]. This is because simulation-based NI-MPC implies virtual black-box (VBB) obfuscation that is known to be impossible for certain classes of functions [13].

[14] Reusable NI-MPC remains secure even when the same setup is used over multiple rounds. On the other hand, non-reusable NI-MPC does not permit to reuse the same setup, i.e., after each round the setup algorithm needs to be executed.

adversary can maliciously compute using the encryption key $\mathsf{ek}_i$ of a corrupted party.

By plugging in our results, we obtain either CPA-1-sided reusable $(n-1)$-robust NI-MPC with $n = O(1)$, or CPA-1-sided reusable 0-robust NI-MPC with $n = \mathsf{poly}(\lambda)$ where the predicate $\mathbb{P}$ of the function $f_{\mathbb{P}}$ is a conjunctions of arbitrary predicates (i.e., $\mathbb{P}(x_1, \ldots, x_n) = \mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)$) with wildcards under the LWE assumption. Note that a CPA-1-sided reusable NI-MPC for $f_{\mathbb{P}}$ where $\mathbb{P}(x_1, \ldots, x_n) = \mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)$ can be used to implement a voting protocol with message passing, i.e., only when each parties' vote $x_i$ satisfies its dedicated set of requirements $\mathbb{P}_i(\cdot)$ (i.e., $\mathbb{P}_i(x_i) = 1$ for every $i \in [n]$) the messages are revealed to all the participants. Until this condition is not satisfied, everything remains secret.

We stress that, nonetheless CPA-1-sided reusability is a weakening of the standard reusability definition, our flavor of reusability is still non-trivial to achieve in the setting of general functions. This is because we can build null iO (and, in turn, witness encryption) [19,31,48] from CPA-1-sided reusable NI-MPC using the same constructions of iO from (standard) reusable NI-MPC, i.e., CPA-1-sided reusable (resp. CPA-1-sided non-reusable) 0-robust (resp. 1-robust) NI-MPC for $n = \mathsf{poly}(\lambda)$ parties (resp. $n = 2$ parties) and general functions implies null iO.

## 1.4 Relation with Witness Encryption (WE)

We observe that both multi-input and multi-key schemes imply witness encryption (WE) [27], if the former support arbitrary predicates (or any predicate that implements a desired NP relation). Brakerski et al. [19] have shown that $n$-input ABE (i.e., predicate inputs can be public), secure in the secret-key setting and without collusions, implies WE for NP and $n$-size witnesses. Similarly, we can build WE from multi-key ABE (i.e., a multi-key scheme where predicate inputs can be public) using a similar construction except that we substitute the multiple inputs with the multiple decryption keys of multi-key ABE. Unfortunately, we cannot use here our constructions of multi-key and multi-input since they only support conjunctions of arbitrary predicates (we stress that CPA-1-sided and CPA-2-sided security are not required for constructing WE).

We also observe that arbitrary predicates are not needed if we consider security under corruptions. Indeed, 2-input ABE for conjunctions of arbitrary predicates $\mathbb{P}(x_1, x_2) = \mathbb{P}_1(x_1) \wedge \mathbb{P}_2(x_2)$ *without wildcards* under 1 corruption and no collusions, implies WE for any relation. Even in this case, our $O(1)$-input scheme under corruptions fails to imply WE. This is because our construction supports conjunctions of arbitrary predicates *each one* having a wildcard (in other words, the wildcard is a trivial witness for any statement). We provide more details in the full version of this work [25].

From these observations, we can identify two plausible approaches that could lead to a construction of WE from standard assumptions: $(i)$ enlarging the class of predicates of our secret-key $n$-input or $n$-key constructions, or $(ii)$ supporting conjunctions of arbitrary predicates (without wildcards) in the setting of 2-input ABE with security under 1 corruption.

## 2 Related Work

Multi-input PE is a special case of multi-input FE [29]. It is well known that so-called compact FE (supporting arbitrary functions) implies multi-input FE [9,15], which in turn implies iO. Constructions of multi-input FE from standard assumptions, in turn, exist for restricted functions [1–4,6,7,16,21,22,24,39,44]. Multi-input PE can also be seen as stronger form of multi-input ABE [19], the difference being that the attributes are not private in the case of ABE. Previously to our work, all (provably secure) constructions of $n$-input ABE with $n > 2$ required iO (the only exception is the concurrent work of Agrawal et al. [8], which we discuss in the next paragraph).

The multi-input and multi-key settings have also been considered in the context of fully-homomorphic encryption [23,40,41].

*Concurrent and Independent Work.* The independent and concurrent work of Agrawal, Yadav, and Yamada [8] proposes two constructions of secret-key (i.e., no corruptions) 2-input key-policy ABE for $\mathsf{NC}^1$ with unbounded collusions (recall that, in the ABE setting, only the secrecy of the messages is guaranteed, i.e., inputs can be public). The first construction is based on LWE and pairings, and it is provably secure in the generic group model. The second construction is based on function-hiding inner-product FE, a variant of the non-falsifiable KOALA knowledge assumption (which is proven to hold under the bilinear generic group model), and LWE. However, this second construction achieves a weaker selective flavor of security in which the adversary has to submit both the challenge and the decryption key queries before the setup phase. Additionally, they propose two heuristic constructions. The first is a 2-input ABE for $\mathsf{P}$ from lattices, and the second is a 3-input ABE for $\mathsf{NC}^1$ from pairings and lattices. However, the security of these heuristic constructions remains unclear.

In comparison, our work directly focuses on the PE setting (i.e., CPA-1-sided security) and provides the first secret-key $n$-input PE that supports $n = \mathsf{poly}(\lambda)$ inputs, with (adaptive) CPA-1-sided security (i.e., secrecy of both inputs and messages) based solely on LWE. However, our construction only supports a restricted class of predicates (i.e., conjunctions of arbitrary predicates with wildcards) and it is secure only in the case of no collusions. Furthermore, differently from [8], we move away from the secret-key setting and propose a second construction of $n$-input PE (still for conjunctions of arbitrary predicates) that supports $n = O(1)$ inputs and can tolerate $n-1$ corruptions (i.e., up to $n-1$ encryption keys can be adaptively revealed by the adversary). Finally, we propose the notion of multi-key PE (not covered in [8]), and give the first construction of

CPA-1-sided secure $n$-key PE for $n = \mathsf{poly}(\lambda)$, with unbounded collusions and still supporting conjunctions of arbitrary predicates, based on LWE.

Regarding the techniques, we highlight that both our work and that of [8] introduce (albeit different) nesting techniques based on lockable obfuscation. In particular, the nesting technique of [8] permits to transform any secret-key $n$-input ABE into a secret-key $n$-input PE (achieving CPA-1-sided security). We stress that their approach only works in the secret-key setting. In contrast, we propose a different nesting technique which yields $n$-input PE for $n = O(1)$ while tolerating $n - 1$ corruptions. It is important to note that our nesting technique is not generic, but it is specifically tailored to work with the class of predicates considered in this work.

Turning to applications, we highlight that the multi-input schemes of [8] fail to imply ME, since their constructions are all in the secret-key setting (whereas ME requires a public-key encryption algorithm). As for NI-MPC, the constructions in [8] can be used to obtain a CPA-1-sided 0-robust reusable NI-MPC for all-or-nothing functions defined over arbitrary predicates, but only in the case of 2 parties (3 parties if we consider also the heuristic constructions).

## 3 Preliminaries

We assume the reader to be familiar with standard cryptographic notation and definitions. The preliminaries can be found in the full version [25].

## 4 Multi-key and Multi-input Predicate Encryption

We provide the formal definitions of multi-key PE and multi-input PE. In the full version [25], we build ME from multi-key PE and CPA-1-sided reusable robust NI-MPC for all-or-nothing functions from multi-input PE.

**Multi-key PE.** Formally, an $n$-key PE with message space $\mathcal{M}$, input space $\mathcal{X}$, and predicate space $\mathcal{P} = \{\mathbb{P}_{v_1,\ldots,v_n}(x)\}_{(v_1,\ldots,v_n)\in\mathcal{V}}$ indexed by $\mathcal{V} = \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$, is composed of the following polynomial-time algorithms:

$\mathsf{Setup}(1^\lambda)$**:** Upon input the security parameter $1^\lambda$ the setup algorithm outputs the master public key $\mathsf{mpk}$ and the $n$ master secret key $(\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$.

$\mathsf{KGen}(\mathsf{msk}_i, v_i)$**:** Let $i \in [n]$. The randomized key generator takes as input the $i$-th master secret key $\mathsf{msk}_i$ and the $i$-th index $v_i \in \mathcal{V}_i$. The algorithm outputs the $i$-th secret decryption key $\mathsf{dk}_{v_i}$ for the predicate index $v_i$.

$\mathsf{Enc}(\mathsf{mpk}, x, m)$**:** The randomized encryption algorithm takes as the master public key $\mathsf{mpk}$, an input $x \in \mathcal{X}$, and a message $m \in \mathcal{M}$. The algorithm produces a ciphertext $c$.

$\mathsf{Dec}(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}, c)$**:** The deterministic decryption algorithm takes as input $n$ secret decryption keys $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$ for the $n$ indexes $(v_1, \ldots, v_n) \in \mathcal{V}$ and a ciphertext $c$. The algorithm outputs a message $m$.

```
$\mathbf{G}^{\mathsf{CPA}\text{-}t\text{-}\mathsf{kPE}}_{\Pi,\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}_1, \ldots, \mathsf{msk}_n) \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$(m^0, m^1, x^0, x^1, \alpha) \leftarrow\$ \mathsf{A}_0^{\mathsf{KGen}(\mathsf{msk}_1,\cdot),\ldots,\mathsf{KGen}(\mathsf{msk}_n,\cdot)}(1^\lambda, \mathsf{mpk})$
$b \leftarrow\$ \{0,1\}, c \leftarrow\$ \mathsf{Enc}(\mathsf{mpk}, x^b, m^b)$
$b' \leftarrow\$ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk}_1,\cdot),\ldots,\mathsf{KGen}(\mathsf{msk}_n,\cdot)}(1^\lambda, c, \alpha)$
If $(b' = b)$: return 1
Else: return 0
```

**Fig. 1.** Game defining CPA-$t$-sided security of $n$-key PE.

Correctness is intuitive: given the decryption keys $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$ for $(v_1, \ldots, v_n) \in \mathcal{V}$, the decryption algorithm returns the message $m$ (encrypted under the input $x$) with overwhelming probability, whenever $\mathbb{P}_{v_1,\ldots,v_n}(x) = 1$. See [25] for the formal definition.

As for security, we adapt the standard CPA-1-sided and CPA-2-sided security of PE to the $n$-key setting. In particular, an adversary (with oracle access to $\mathsf{KGen}(\mathsf{msk}_i, \cdot)$ for $i \in [n]$) cannot distinguish between $\mathsf{Enc}(\mathsf{mpk}, x^0, m^0)$ and $\mathsf{Enc}(\mathsf{mpk}, x^1, m^1)$ except with non-negligible probability. When considering CPA-1-sided security, the adversary is valid only if it cannot decrypt the challenge ciphertext, i.e., it asks to the $n$ key generation oracles indexes $(v_1, \ldots, v_n)$ such that $\mathbb{P}_{v_1,\ldots,v_n}(x^0) = \mathbb{P}_{v_1,\ldots,v_n}(x^1) = 0$. Analogously, the CPA-2-sided security captures the indistinguishability of $\mathsf{Enc}(\mathsf{mpk}, x^0, m^0)$ and $\mathsf{Enc}(\mathsf{mpk}, x^1, m^1)$ even when the adversary can decrypt the challenge ciphertext, i.e., $\mathbb{P}_{v_1,\ldots,v_n}(x^0) = \mathbb{P}_{v_1,\ldots,v_n}(x^1) = 1$ and $m^0 = m^1$. These security definitions are formalized below.

**Definition 1 (CPA-1-sided and CPA-2-sided security of $n$-key PE).** *Let $t \in [2]$. We say that a $n$-key PE $\Pi$ is CPA-$t$-sided secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:*

$$\left| \mathbb{P}\left[\mathbf{G}^{\mathsf{CPA}\text{-}t\text{-}\mathsf{kPE}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

*where game $\mathbf{G}^{\mathsf{CPA}\text{-}t\text{-}\mathsf{kPE}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 1. Adversary $\mathsf{A}$ is called valid if $\forall v_1 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)}, \ldots, \forall v_n \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_n,\cdot)}$, we have*

$$\textbf{\textit{Case}}\ t = 1\textbf{:}\ \mathbb{P}_{v_1,\ldots,v_n}(x^0) = \mathbb{P}_{v_1,\ldots,v_n}(x^1) = 0.$$
$$\textbf{\textit{Case}}\ t = 2\textbf{:}\ \textit{Either}\ \mathbb{P}_{v_1,\ldots,v_n}(x^0) = \mathbb{P}_{v_1,\ldots,v_n}(x^1) = 0$$
$$\textit{or}\ \mathbb{P}_{v_1,\ldots,v_n}(x^0) = \mathbb{P}_{v_1,\ldots,v_n}(x^1) \wedge m^0 = m^1.$$

**Multi-input PE.** Formally, an $n$-input PE with message space $\mathcal{M} = \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$, input space $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$, and predicate space $\mathcal{P}$, is composed of the following polynomial-time algorithms:

$\mathsf{Setup}(1^\lambda)$**:** Upon input the security parameter $1^\lambda$ the setup algorithm outputs the encryption keys $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ and the master secret key $\mathsf{msk}$.

$\mathsf{KGen}(\mathsf{msk}, \mathbb{P})$**:** The randomized key generator takes as input the master secret key $\mathsf{msk}$ and a predicate $\mathbb{P} \in \mathcal{P}$. The algorithm outputs a secret decryption key $\mathsf{dk}_{\mathbb{P}}$ for predicate $\mathbb{P}$.

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$**:** Let $i \in [n]$. The randomized encryption algorithm takes as input an encryption key $\mathsf{ek}_i$, an input $x_i \in \mathcal{X}_i$, and a message $m_i \in \mathcal{M}_i$. The algorithm produces a ciphertext $c_i$ linked to $x_i$.

$\mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, c_1, \ldots, c_n)$**:** The deterministic decryption algorithm takes as input a secret decryption key $\mathsf{dk}_{\mathbb{P}}$ for predicate $\mathbb{P} \in \mathcal{P}$ and $n$ ciphertexts $(c_1, \ldots, c_n)$. The algorithm outputs $n$ messages $(m_1, \ldots, m_n)$.

Correctness states that ciphertexts $(c_1, \ldots, c_n)$, each linked to an input $x_i$, correctly decrypt with overwhelming probability if $\mathbb{P}(x_1, \ldots, x_n) = 1$. See the full version of this work [25] for the formal definition.

*Security with and without Corruptions.* The CPA-1-sided and CPA-2-sided security of $n$-input PE capture the infeasibility in distinguishing between ciphertexts $(\mathsf{Enc}(\mathsf{ek}_1, x_1^0, m_1^0), \ldots, \mathsf{Enc}(\mathsf{ek}_n, x_n^0, m_n^0))$ and $(\mathsf{Enc}(\mathsf{ek}_1, x_1^1, m_1^1), \ldots, \mathsf{Enc}(\mathsf{ek}_n, x_n^1, m_n^1))$. This is modeled by an adversary having oracle access to a key generation oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$ (allowing it to get decryption keys $\mathsf{dk}_{\mathbb{P}}$ on predicates of its choice) and $n$ encryption oracles $\mathsf{Enc}(\mathsf{ek}_1, \cdot, \cdot), \ldots, \mathsf{Enc}(\mathsf{ek}_n, \cdot, \cdot)$ (allowing it to get encryptions of arbitrary messages and inputs). Differently from the $n$-key setting, we consider different models of security with respect to whether the encryption keys are secret (i.e., no corruptions) or public/leaked (i.e., the adversary has the possibility to get one or more encryption keys of its choice). The corruption of an encryption key is captured by giving access to a corruption oracle $\mathsf{Corr}(\cdot)$ to the adversary that, on input $i \in [n]$, it returns $\mathsf{ek}_i$. Intuitively, the knowledge of $\mathsf{ek}_i$ impacts the validity condition that the adversary must satisfy (e.g., the challenge ciphertext cannot be decrypted). Indeed, $\mathsf{ek}_i$ would allow the adversary to produce arbitrary $i$-th ciphertexts on arbitrary $i$-th inputs $x_i$ and potentially decrypt part of the challenge ciphertext. Concretely, as for CPA-1-sided security, the validity of the adversary can be defined as follows:

– *No corruptions (a.k.a. the secret-key setting).* If all the encryption keys $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ are secret the validity conditions of CPA-1-sided security is straightforward. It intuitively states that for every $\mathsf{dk}_{\mathbb{P}}$ (obtained through oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$) and any tuple of ciphertexts $(c_1, \ldots, c_n)$ (each linked to an input $x_i$) obtained through the interleaving of part of the challenge ciphertext with the ciphertexts generated by invoking oracles $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i \in [n]}$, we must have that $\mathbb{P}(x_1, \ldots, x_n) = 0$ (otherwise part of the challenge ciphertext can be decrypted).

```
$\mathbf{G}_{\Pi,\mathsf{A}}^{\ell\text{-}\mathsf{CPA}\text{-}t\text{-}\mathsf{iPE}}(\lambda)$
$(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$
$((m_i^0)_{i\in[n]}, (m_i^1)_{i\in[n]}, (x_i^0)_{i\in[n]}, (x_i^1)_{i\in[n]}, \alpha) \leftarrow\!\$\ \mathsf{A}_0^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Corr}(\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[n]}}(1^\lambda)$
$b \leftarrow\!\$\ \{0,1\}, c_1 \leftarrow\!\$\ \mathsf{Enc}(\mathsf{ek}_1, x_1^b, m_1^b), \ldots, c_n \leftarrow\!\$\ \mathsf{Enc}(\mathsf{ek}_n, x_n^b, m_n^b)$
$b' \leftarrow\!\$\ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Corr}(\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[n]}}(1^\lambda, c_1, \ldots, c_n, \alpha)$
If $(b' = b)$: return 1
Else: return 0
```

**Fig. 2.** Game defining CPA-$t$-sided security of $n$-input PE in the $\ell$-corruptions setting. Oracle $\mathsf{Corr}(j)$ returns $\mathsf{ek}_j$ for $j \in [n]$.

– *With corruptions.* If some of the encryption keys are known by the adversary (i.e., obtained through the corruption oracle $\mathsf{Corr}(\cdot)$) then the validity condition now changes according to which keys have been obtained. This is because the adversary can now autonomously compute arbitrary ciphertext (for a particular slot $i$) using the leaked $i$-th encryption key $\mathsf{ek}_i$. Taking into account this observation, the validity of CPA-1-sided security *with corruptions* says that any tuple of ciphertexts $(c_1, \ldots, c_n)$ that can be obtained by interleaving part of the challenge ciphertexts with both the ones generated through oracles $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i\in[n]}$ and the ones that can be autonomously generated using the leaked encryption keys, we must have that $\mathbb{P}(x_1, \ldots, x_n) = 0$.

The validity of CPA-2-sided security (with and without corruptions) can be easily obtained by adapting the above discussion. Below, we provide the formal definition.

**Definition 2 ($\ell$-Corruptions CPA-1-sided and CPA-2-sided security of $n$-input PE).** *Let $t \in [2]$. We say that an $n$-input PE $\Pi$ is CPA-$t$-sided secure in the $\ell$-corruptions setting if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:*

$$\left| \mathbb{P}\big[\mathbf{G}_{\Pi,\mathsf{A}}^{\ell\text{-}\mathsf{CPA}\text{-}t\text{-}\mathsf{iPE}}(\lambda) = 1\big] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

*where game $\mathbf{G}_{\Pi,\mathsf{A}}^{\ell\text{-}\mathsf{CPA}\text{-}t\text{-}\mathsf{iPE}}(\lambda)$ is depicted in Fig. 2. Let $\mathcal{Q}_i = \{x | \exists (x, m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_i,\cdot,\cdot)}\}$ for $i \in [n] \setminus \mathcal{Q}_{\mathsf{Corr}}$ and $\mathcal{Q}_i = \mathcal{X}_i$ for $i \in \mathcal{Q}_{\mathsf{Corr}}$. Moreover, let $\mathcal{Q}_i^d$ (for $d \in \{0,1\}$) be the ordered list composed of the predicate inputs $\mathcal{Q}_i$ and the challenge input $x_i^d$, i.e., $\mathcal{Q}_i^d = \{x_i^{(1,d)}, \ldots, x_i^{(k_i,d)}, x_i^{(k_i+1,d)} = x_i^d\}$ where $k_i = |\mathcal{Q}_i|$ and $x^{(j,d)} \in \mathcal{Q}_i$ for $j \in [k_i]$.[15] Adversary $\mathsf{A}$ is called valid if $|\mathcal{Q}_{\mathsf{Corr}}| \leq \ell$ and*

[15] Observe that $\mathcal{Q}_i^0$ and $\mathcal{Q}_i^1$ are identical except for the last element.

$\forall \mathbb{P} \in \mathcal{Q}_{\mathsf{KGen}}$, $\forall j \in [n]$, $\forall i_1 \in [k_1+1], \ldots, \forall i_n \in [k_n+1]$, *we have*

***Case*** $t = 1$***:*** $\mathbb{P}(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) =$
$\mathbb{P}(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) = 0.$

***Case*** $t = 2$***:*** *Either*

$$\mathbb{P}(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) =$$
$$\mathbb{P}(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) = 0$$

*or*

$$\mathbb{P}(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) =$$
$$\mathbb{P}(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) \wedge m_j^0 = m_j^1.$$

Through the paper, for $t \in [2]$, we say that $\Pi$ is CPA-$t$-sided secure in the $\ell$-corruptions setting and *without collusions* if $|\mathcal{Q}_{\mathsf{KGen}}| = 1$ (i.e., the adversary asks for a single decryption key). If $|\mathcal{Q}_{\mathsf{Corr}}| = 0$ (i.e., no corruptions), we say that $\Pi$ is CPA-$t$-sided secure in the *secret-key setting.* In case of both restrictions, we say that $\Pi$ is CPA-$t$-sided secure in the *secret-key setting* and *without collusions* (i.e., $|\mathcal{Q}_{\mathsf{Corr}}| = 0$ and $|\mathcal{Q}_{\mathsf{KGen}}| = 1$).

*Remark 1 (Relation with multi-key PE).* In the full version of this work [25], we show that CPA-$t$-sided secure $(n+1)$-input PE tolerating 1 corruption, naturally implies CPA-$t$-sided secure $n$-key PE.[16] We stress that such a relation holds only if the $(n+1)$-input PE supports arbitrary predicates. On the other hand, if we consider restricted classes of predicates (as studied in this work), the above implication does not to hold anymore, making multi-input and multi-key PE incomparable.[17]

Also, we discuss the relation between the multi-key and multi-input settings when considering a weaker definition of security. In particular, if we drop the secrecy of the predicate inputs, i.e., only the the messages remain secret (which is equivalent to ABE), then we can show that multi-key ABE implies multi-input ABE only in the presence of no corruptions.

## 5 Constructions

In this section, we give different constructions of multi-key and multi-input PE (see also Sect. 1.2) for predicates $\mathbb{P}(x_1, \ldots, x_n) = \mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)$.

[16] If we restrict the $n$-key PE's encryption algorithm to be secret-key (i.e., $\mathsf{Enc}(\mathsf{ek}, \cdot, \cdot)$ where $\mathsf{ek}$ is kept secret) then we can start from a secret-key $(n+1)$-input PE, i.e., 0 corruptions.

[17] This is also reflected by the results achieved in this paper. For example, our multi-key PE construction for conjunctions of arbitrary predicates tolerates unbounded collusions whereas our multi-input PE constructions (for the same class of predicates with wildcards) are significantly more complex and are secure only in the case of no collusions.

```
$\mathbb{C}_c(\mathsf{dk}_1, \ldots, \mathsf{dk}_n)$

Initialize: $c_n = c$
For $i$ from $n$ to 1 do: $\mathsf{Dec}_i(\mathsf{dk}_i, c_i) = c_{i-1}$
return $c_0$
```

**Fig. 3.** Definition of the circuit $\mathbb{C}_c$ of Construction 1.

In particular, in Sect. 5.1 we give a construction of $n$-key PE from single-input PE and lockable obfuscation for $n = \mathsf{poly}(\lambda)$. This construction is secure against unbounded collusions.

In Sect. 5.2, we give a construction of $O(1)$-input PE, that is CPA-1-side secure without collusions and in the $(n-1)$-corruptions setting, from single-input PE, lockable obfuscation, and PKE. It leverages a new nesting execution technique of (lockable obfuscated) circuits. Our secret-key $n$-input PE construction for $n = \mathsf{poly}(\lambda)$ is deferred to full version [25].

Both multi-input constructions support conjunctions of arbitrary predicates with wildcards, i.e., for every $i \in [n]$, there exists (possibly unique) a wildcard $x_i^\star$ such that for every $i$-th predicate $\mathbb{P}_i$ we have $\mathbb{P}_i(x_i^\star) = 1$ (in [25] we discuss how to remove the wildcard when no corruptions are in place).

Also, our constructions are generic and achieve CPA-2-sided security if the underlying single-input PE is CPA-2-sided secure (our CPA-2-sided secure secret-key multi-input PE construction (see [25]) supports $n = O(\log(\lambda))$).

## 5.1 Multi-key PE from PE and Lockable Obfuscation

**Construction 1.** *Consider the following primitives:*

1. *For $i \in [n]$, a PE scheme $\mathsf{PE}_i = (\mathsf{Setup}_i, \mathsf{KGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ with message space $\mathcal{M}_i$, input space $\mathcal{X}_i$, and predicate space $\mathcal{P}_i = \{\mathbb{P}_v(x)\}_{v \in \mathcal{V}_i}$ indexed by $\mathcal{V}_i$. Without loss of generality, we assume that $\mathsf{PE}_i$ has ciphertext space $\mathcal{Y}_i$, $\mathcal{M}_1 = \{0,1\}^{m(\lambda)}$, and $\mathcal{M}_i = \mathcal{Y}_{i-1}$ for every $i \in [n] \setminus \{1\}$. In order to do not incur into an exponential ciphertext growth (e.g., for $n = \mathsf{poly}(\lambda)$), each $i$-th PE scheme must have a ciphertext expansion of $\mathsf{poly}(\lambda) + |m_i|$ where $|m_i|$ is the length of the messages $m_i \in \mathcal{M}_i$ supported by the $i$-th PE scheme (this can be obtained generically from any PE scheme by leveraging hybrid encryption, i.e., $\mathsf{Enc}_i(\mathsf{mpk}, x, s)||\mathsf{PRG}(s) \oplus m_i$ where $s \leftarrow_\$ \{0,1\}^\lambda$).*
2. *A lockable obfuscation scheme $\mathsf{LOBF} = (\mathsf{Obf}, \mathsf{Eval})$ with message space $\mathcal{M}$ for the family of circuits $\mathcal{C}_{n,s,d}(\lambda) = \{\mathbb{C}_c\}$ as defined in Fig. 3, where $n(\lambda)$, $s(\lambda)$, $d(\lambda)$ depends on the schemes $\mathsf{PE}_1, \ldots, \mathsf{PE}_n$ used, and the circuits $\mathcal{C}_{n,s,d}(\lambda)$.*

*We build a $n$-key PE scheme $\Pi$ with message space $\mathcal{M}$, input space $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$, and predicate space $\mathcal{P} = \{\mathbb{P}_{v_1,\ldots,v_n}(x_1,\ldots,x_n) = \mathbb{P}_{v_1}(x_1) \wedge \ldots \wedge \mathbb{P}_{v_n}(x_n)\}_{(v_1,\ldots,v_n) \in \mathcal{V}}$ indexed by $\mathcal{V} = \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$ (and $\mathbb{P}_{v_i} \in \mathcal{P}_i$ for $i \in [n]$), as follows:*

$\mathsf{Setup}(1^\lambda)$**:** *Upon input the security parameter* $1^\lambda$ *the randomized setup algorithm outputs* $\mathsf{mpk} = (\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n)$ *and* $\mathsf{msk}_1, \ldots, \mathsf{msk}_n$ *where* $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow\!\!\$\ \mathsf{Setup}_i(1^\lambda)$ *for* $i \in [n]$.

$\mathsf{KGen}(\mathsf{msk}_i, v_i)$**:** *Let* $i \in [n]$. *Upon input the* $i$*-th master secret key* $\mathsf{msk}_i$ *and the* $i$*-th predicate index* $v_i \in \mathcal{V}_i$, *the randomized key generator outputs* $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_1, \mathbb{P}_{v_i})$ *where* $\mathbb{P}_{v_i} \in \mathcal{P}_i$.

$\mathsf{Enc}(\mathsf{mpk}, x, m)$**:** *Upon input the master public key* $\mathsf{mpk} = (\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n)$, *an input* $x = (x_1, \ldots, x_n) \in \mathcal{X}$, *and a message* $m \in \mathcal{M}$, *the randomized encryption proceeds as follows:*
1. *Sample* $y \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ *and let* $c_0 = y$.
2. *For* $i \in [n]$, *compute* $c_i \leftarrow\!\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x_i, c_{i-1})$.

*Finally, it outputs* $c = \widetilde{\mathbb{C}}$ *where* $\widetilde{\mathbb{C}} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_n}, y, m)$.

$\mathsf{Dec}(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}, c)$**:** *Upon input* $n$ *decryption keys* $\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}$ *and a ciphertext* $c = \widetilde{\mathbb{C}}$, *the deterministic decryption algorithm outputs* $m = \mathsf{Eval}(\widetilde{\mathbb{C}}, (\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}))$.

Correctness follows from the correctness of the underlying schemes. We establish the following result whose proof is deferred to full version [25].

**Theorem 4.** *Let* $n = \mathsf{poly}(\lambda)$, $\mathsf{PE}_1, \ldots, \mathsf{PE}_n$ *and* $\mathsf{LOBF}$ *be as above. If* $\mathsf{LOBF}$ *is secure and*

1. *each* $\mathsf{PE}_1, \ldots, \mathsf{PE}_n$ *is CPA secure, then the* $n$*-key PE scheme* $\Pi$ *from Construction 1 is CPA-1-sided secure (Definition 1).*
2. *each* $\mathsf{PE}_1, \ldots, \mathsf{PE}_n$ *is CPA-2-sided secure, then the* $n$*-key PE scheme* $\Pi$ *from Construction 1 is CPA-2-sided secure (Definition 1).*

We stress that CPA secure single-input PE (see the above theorem) guarantees only the secrecy of the message (whereas predicate inputs can be public). This is equivalent to the notion of single-input ABE.

## 5.2 Multi-input PE from PE, Lockable Obfuscation and PKE

*Corruption Setting.* We present our construction of $n$-input PE that is CPA-1-sided secure in the $(n-1)$-corruptions setting without collusions. This construction handles constant-arity (i.e., $n \in O(1)$) since the decryption running time is $O(n^n)$. It is based on CPA secure single-input PE, lockable obfuscation, and PKE and it leverages the nested execution technique described in Sect. 1.2. Also, the same construction achieves CPA-2-sided security if the initial single-input PE is CPA-2-sided secure.

**Construction 2** ($n$-input PE in the corruption setting). *Consider the following primitives:*

1. *A PE scheme* $\mathsf{PE} = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *with message space* $\mathcal{M}_1 = \{0,1\}^{m_3(\lambda)+m_4(\lambda)}$, *input space* $\mathcal{X}_1 = \mathcal{X}_{1,1} \times \ldots \times \mathcal{X}_{1,n}$, *and predicate space* $\mathcal{P}_1 = \{\mathbb{P}(x_1, \ldots, x_n)\} = \{\mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)\}$. *Without loss of generality, we assume that* $\mathsf{PE}$ *has ciphertext space* $\mathcal{Y}_1$ *and there exists a (single) wildcard input* $(x_1^\star, \ldots, x_n^\star) \in \mathcal{X}_1$ *such that* $\forall (\mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)) \in \mathcal{P}_1, \forall i \in [n], \mathbb{P}_i(x_i^\star) = 1$.

```
$\mathbb{C}^{\mathsf{in}}_{c,\mathsf{sk},i}(\mathbb{C}_1, \ldots, \mathbb{C}_{n-2}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{dk}_{\mathbb{P}})$

Initialize:
  $c_n = c, \mathsf{sk}'_i = \mathsf{sk}, \mathbb{C}_{n-1} = \perp, k = \perp, \forall j \in [n] \setminus \{i\},\ \mathsf{sk}'_j = \mathsf{sk}_j$
  If $\exists w \in [n-2]$ such that $\mathbb{C}_w \neq \perp$ and $\mathbb{C}_{w+1} = \perp$: $k = w$
end initialize.
If $k \neq \perp$ do:  // If k = ⊥, no circuit to execute.
  // Execute each circuit received in input in order to retrieve the related secret key.
  For $t \in [k]$ do:
    $\mathsf{Eval}_3(\mathbb{C}_t, (\mathbb{C}_{t+1}, \ldots, \mathbb{C}_k, \overbrace{\perp, \ldots, \perp}^{n-2+t-k}, \mathsf{sk}'_1, \ldots, \mathsf{sk}'_n, \mathsf{dk}_{\mathbb{P}})) = r$
    If $r = \perp$: return $\perp$
    Else: $\mathsf{sk}'_h = \overline{\mathsf{sk}}$ where $r = (\overline{\mathsf{sk}}, h)$   // Save the secret key returned by $\mathbb{C}_t$.
  end for.
end if.
// At this point, all secret keys are known.
For $j$ from $n$ to 1 do: $\mathsf{Dec}_{2,j}(\mathsf{sk}'_j, c_j) = c_{j-1}$
$\mathsf{Dec}_1(\mathsf{dk}_{\mathbb{P}}, c_0) = v$
If $v = \perp$: return $\perp$
Else: return $y_i^{\mathsf{in}}$ where $v = (y_i^{\mathsf{in}}, y_i^{\mathsf{out}})$

$\mathbb{C}^{\mathsf{out}}_{c,\mathsf{sk},i}(\mathbb{C}_1, \ldots, \mathbb{C}_{n-1}, \mathsf{dk}_{\mathbb{P}})$

Initialize: $c_n = c, \mathsf{sk}'_i = \mathsf{sk}, \forall j \in [n] \setminus \{i\},\ \mathsf{sk}'_j = \perp$
// Execute each circuit received in input in order to retrieve the related secret key.
For $t$ from 1 to $n-1$ do:
  $\mathsf{Eval}_3(\mathbb{C}_t, (\mathbb{C}_{t+1}, \ldots, \mathbb{C}_{n-1}, \overbrace{\perp, \ldots, \perp}^{t-1}, \mathsf{sk}'_1, \ldots, \mathsf{sk}'_n, \mathsf{dk}_{\mathbb{P}})) = r$
  If $r = \perp$: return $\perp$
  Else: $\mathsf{sk}'_h = \overline{\mathsf{sk}}$ where $r = (\overline{\mathsf{sk}}, h)$   // Save the secret key returned by $\mathbb{C}_t$.
end for.
// At this point, all secret keys are known.
For $j$ from $n$ to 1 do: $\mathsf{Dec}_{2,j}(\mathsf{sk}'_j, c_j) = c_{j-1}$
$\mathsf{Dec}_1(\mathsf{dk}_{\mathbb{P}}, c_0) = v$
If $v = \perp$: return $\perp$
Else: return $y_i^{\mathsf{out}}$ where $v = (y_i^{\mathsf{in}}, y_i^{\mathsf{out}})$
```

**Fig. 4.** Definitions of the circuits $\mathbb{C}^{\mathsf{in}}_{c,\mathsf{sk},i}$ and $\mathbb{C}^{\mathsf{out}}_{c,\mathsf{sk},i}$ supported by the lockable obfuscation schemes $\mathsf{LOBF}_3$ and $\mathsf{LOBF}_4$ of Construction 2.

2. *For $i \in [n]$, a PKE scheme* $\mathsf{PKE}_{2,i} = (\mathsf{KGen}_{2,i}, \mathsf{Enc}_{2,i}, \mathsf{Dec}_{2,i})$ *with message space $\mathcal{M}_{2,i}$. Without loss of generality, we assume that* $\mathsf{PKE}_i$ *has ciphertext space $\mathcal{Y}_{2,i}$ and secret-key space $\mathcal{K}_{2,i}$. Moreover, we assume that $\mathcal{M}_{2,1} = \mathcal{Y}_1$, and $\mathcal{M}_{2,i} = \mathcal{Y}_{2,i-1}$ for every $i \in [n] \setminus \{1\}$.*
3. *A lockable obfuscation scheme* $\mathsf{LOBF}_3 = (\mathsf{Obf}_3, \mathsf{Eval}_3)$ *with message space $\mathcal{M}_3 = (\mathcal{K}_{2,1} \cup \ldots \cup \mathcal{K}_{2,n}) \times \{0,1\}^{\lfloor \log_2(n) \rfloor + 1}$ for the family of circuits $\mathcal{C}^{\mathsf{in}}_{n_3,s_3,d_3}(\lambda) = \{\mathbb{C}^{\mathsf{in}}_{c,\mathsf{sk},i}\}$ defined in Fig. 4, where $n_3(\lambda)$, $s_3(\lambda)$, $d_3(\lambda)$ depends on the schemes* $\mathsf{PE}, \mathsf{PKE}_{2,1}, \ldots, \mathsf{PKE}_{2,n}$ *used, and the circuits $\mathcal{C}^{\mathsf{in}}_{n_3,s_3,d_3}(\lambda)$.*
4. *A lockable obfuscation scheme* $\mathsf{LOBF}_4 = (\mathsf{Obf}_4, \mathsf{Eval}_4)$ *with message space $\mathcal{M}_4$ for the family of circuits $\mathcal{C}^{\mathsf{out}}_{n_4,s_4,d_4}(\lambda) = \{\mathbb{C}^{\mathsf{out}}_{c,\mathsf{sk},i}\}$ defined in Fig. 4, where $n_4(\lambda)$, $s_4(\lambda)$, $d_4(\lambda)$ depends on the schemes* $\mathsf{PE}, \mathsf{PKE}_{2,1}, \ldots, \mathsf{PKE}_{2,n}, \mathsf{LOBF}_3$ *used, and the circuits $\mathcal{C}^{\mathsf{out}}_{n_4,s_4,d_4}(\lambda)$.*

*We build a* $n$*-input PE scheme with message space* $\mathcal{M} = \overbrace{\mathcal{M}_4 \times \ldots \times \mathcal{M}_4}^{n}$*, input space* $\mathcal{X} = \mathcal{X}_1$*, and predicate space* $\mathcal{P} = \mathcal{P}_1 = \{\mathbb{P}(x_1, \ldots, x_n)\} = \{\mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)\}$ *with wildcard (i.e., there exists a (single) wildcard* $(x_1^\star, \ldots, x_n^\star) \in \mathcal{X}$ *such that* $\forall (\mathbb{P}_1(x_1) \wedge \ldots \wedge \mathbb{P}_n(x_n)) \in \mathcal{P}$*,* $\forall i \in [n]$*,* $\mathbb{P}_i(x_i^\star) = 1$*), as follows:*

$\mathsf{Setup}(1^\lambda)$**:** *Upon input the security parameter* $1^\lambda$ *the randomized setup algorithm outputs* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ *and* $\mathsf{msk}$ *where* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$*,* $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{sk}_i, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$*, and* $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\!\$\ \mathsf{KGen}_{2,i}(1^\lambda)$ *for* $i \in [n]$*.*

$\mathsf{KGen}(\mathsf{msk}, \mathbb{P})$**:** *Upon input the master secret key* $\mathsf{msk}$ *and a predicate* $\mathbb{P} \in \mathcal{P}$*, the randomized key generator algorithm outputs* $\mathsf{dk}_\mathbb{P} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P})$*.*

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$**:** *Let* $i \in [n]$*. Upon input an encryption key* $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{sk}_i, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$*, an input* $x_i \in \mathcal{X}_{1,i}$*, and a message* $m_i \in \mathcal{M}_4$*, the randomized encryption algorithm samples* $(y_i^{\mathsf{in}}, y_i^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$ *and proceeds as follows:*

1. *Compute* $c_i^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_i^{\mathsf{in}}, y_i^{\mathsf{out}}))$ *where* $x_j = x_j^\star$ *for* $j \in [n] \setminus \{i\}$*.*
2. *For* $j \in [n]$*, compute* $c_i^{(j)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,j}(\mathsf{pk}_j, c_i^{(j-1)})$*.*

*Finally, it outputs* $c_i = (\widetilde{\mathbb{C}}_i^{\mathsf{out}}, \widetilde{\mathbb{C}}_i^{\mathsf{in}})$*, where* $\widetilde{\mathbb{C}}_i^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_i^{(n)}, \mathsf{sk}_i, i}, y_i^{\mathsf{out}}, m_i)$ *and* $\widetilde{\mathbb{C}}_i^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_i^{(n)}, \mathsf{sk}_i, i}, y_i^{\mathsf{in}}, (\mathsf{sk}_i, i))$*.*

$\mathsf{Dec}(\mathsf{dk}_\mathbb{P}, c_1, \ldots, c_n)$**:** *Upon input a decryption key* $\mathsf{dk}_\mathbb{P}$ *for predicate* $\mathbb{P} \in \mathcal{P}$*, and* $n$ *ciphertexts* $(c_1, \ldots, c_n)$ *such that* $c_i = (\widetilde{\mathbb{C}}_i^{\mathsf{out}}, \widetilde{\mathbb{C}}_i^{\mathsf{in}})$ *for* $i \in [n]$*. The deterministic decryption algorithm returns* $(m_1, \ldots, m_n)$ *where* $m_i = \mathsf{Eval}_4(\widetilde{\mathbb{C}}_i^{\mathsf{out}}, (\widetilde{\mathbb{C}}_1^{\mathsf{in}}, \ldots, \widetilde{\mathbb{C}}_{i-1}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{i+1}^{\mathsf{in}}, \ldots, \widetilde{\mathbb{C}}_n^{\mathsf{in}}, \mathsf{dk}_\mathbb{P}))$ *for* $i \in [n]$*.*

Correctness follows from the one of the underlying primitives (see also Fig. 4 for the definitions of $\mathbb{C}^{\mathsf{in}}_{c,\mathsf{sk},i}$ and $\mathbb{C}^{\mathsf{out}}_{c,\mathsf{sk},i}$). Moreover, decryption is polynomial time when $n \in O(1)$. Below, we establish the following result whose proof is deferred to full version [25].

**Theorem 5.** *Let* $n = O(1)$*,* $\mathsf{PE}$*,* $\mathsf{PKE}_{2,1}, \ldots, \mathsf{PKE}_{2,n}$*,* $\mathsf{LOBF}_3$*, and* $\mathsf{LOBF}_4$ *be as above. If each* $\mathsf{PKE}_{2,i}$ *(for* $i \in [n]$*) is CPA secure, both* $\mathsf{LOBF}_3$ *and* $\mathsf{LOBF}_4$ *are secure, and*

1. $\mathsf{PE}$ *is CPA secure without collusions, then the* $n$*-input PE scheme* $\Pi$ *from Construction 2 is CPA-1-sided secure in the* $(n-1)$*-corruptions setting without collusions (Definition 2).*
2. $\mathsf{PE}$ *is CPA-2-sided secure without collusions, then the* $n$*-input PE scheme* $\Pi$ *from Construction 2 is CPA-2-sided secure in the* $(n-1)$*-corruptions setting without collusions (Definition 2).*

As for Theorem 4, CPA secure single-input PE (see the above theorem) guarantees only the secrecy of the message (whereas predicate inputs can be public). This is equivalent to the notion of single-input ABE.

**Acknowledgements.** The authors would like to thank the anonymous reviewers for useful feedback. The first author was supported by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the second and the fourth author were partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU and by Sapienza University under the project SPECTRA; the third author was partially supported by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038 and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

## References

1. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_19
2. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 128–157. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_5
3. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_20
4. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_21
5. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_2
6. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 208–238. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84259-8_8
7. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption: Stronger security, broader functionality. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, pp. 711–740. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-22318-1_25
8. Agrawal, S., Yadav, A., Yamada, S.: Multi-input attribute based encryption and predicate encryption. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, pp. 590–621. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15802-5_21
9. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_15

10. Ateniese, G., Francati, D., Nuñez, D., Venturi, D.: Match me if you can: matchmaking encryption and its applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 701–731. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_24
11. Ateniese, G., Francati, D., Nuñez, D., Venturi, D.: Match me if you can: matchmaking encryption and its applications. J. Cryptol. **34**(3), 1–50 (2021). https://doi.org/10.1007/s00145-021-09381-4
12. Attrapadung, N.: Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_31
13. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
14. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 387–404. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_22
15. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th FOCS, pp. 171–190. IEEE Computer Society Press (2015). https://doi.org/10.1109/FOCS.2015.20
16. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_19
17. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
18. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Factoring and pairings are not necessary for IO: circular-secure LWE suffices. In: ICALP 2022. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ICALP.2022.28
19. Brakerski, Z., Jain, A., Komargodski, I., Passelègue, A., Wichs, D.: Non-trivial witness encryption and null-iO from standard assumptions. In: Catalano, D., De Prisco, R. (eds.) SCN 2018. LNCS, vol. 11035, pp. 425–441. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_23
20. Chen, J., Li, Y., Wen, J., Weng, J.: Identity-based matchmaking encryption from standard assumptions. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, pp. 394–422. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22969-5_14
21. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 703–732. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_24
22. Ciampi, M., Siniscalchi, L., Waldner, H.: Multi-client functional encryption for separable functions. In: Garay, J.A. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 724–753. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75245-3_26
23. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 630–656. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_31

24. Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the $k$-linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 245–277. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_9
25. Francati, D., Friolo, D., Malavolta, G., Venturi, D.: Multi-key and multi-input predicate encryption from learning with errors. Cryptology ePrint Archive (2022)
26. Francati, D., Guidi, A., Russo, L., Venturi, D.: Identity-based matchmaking encryption without random oracles. In: Adhikari, A., Küsters, R., Preneel, B. (eds.) INDOCRYPT 2021. LNCS, vol. 13143, pp. 415–435. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92518-5_19
27. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 467–476. ACM Press (2013). https://doi.org/10.1145/2488608.2488667
28. Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. In: Khuller, S., Williams, V.V. (eds.) STOC 2021: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, 21–25 June 2021, pp. 736–749. ACM (2021). https://doi.org/10.1145/3406325.3451070
29. Goldwasser, S., et al.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_32
30. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_25
31. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: Umans, C. (ed.) 58th FOCS, pp. 612–621. IEEE Computer Society Press (2017). https://doi.org/10.1109/FOCS.2017.62
32. Halevi, S., Ishai, Y., Jain, A., Komargodski, I., Sahai, A., Yogev, E.: Non-interactive multiparty computation without correlated randomness. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 181–211. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70700-6_7
33. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Sudan, M. (ed.) ITCS 2016, pp. 157–168. ACM (2016). https://doi.org/10.1145/2840728.2840760
34. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: computing without simultaneous interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_8
35. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Khuller, S., Williams, V.V. (eds.) STOC 2021: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, 21–25 June 2021, pp. 60–73. ACM (2021). https://doi.org/10.1145/3406325.3451093
36. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in $NC^0$. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 670–699. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-06944-4_23
37. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9

38. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
39. Libert, B., Ţiţiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_18
40. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Karloff, H.J., Pitassi, T. (eds.) 44th ACM STOC, pp. 1219–1234. ACM Press (2012). https://doi.org/10.1145/2213977.2214086
41. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_26
42. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_11
43. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_35
44. Tomida, J.: Tightly secure inner product functional encryption: multi-input and function-hiding constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 459–488. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_16
45. Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_14
46. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_26
47. Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 127–156. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77883-5_5
48. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: Umans, C. (ed.) 58th FOCS, pp. 600–611. IEEE Computer Society Press (2017). https://doi.org/10.1109/FOCS.2017.61