J Cryptol (2024) 37:24
https://doi.org/10.1007/s00145-024-09504-7

Journal of CRYPTOLOGY

Research Article

Multi-key and Multi-input Predicate Encryption (for Conjunctions) from Learning with Errors*

Danilo Francati
Aarhus University, Aarhus, Denmark
dfrancati@cs.au.dk

Daniele Friolo
Sapienza University of Rome, Rome, Italy

Giulio Malavolta
Bocconi University, Milan, Italy

Daniele Venturi
Sapienza University of Rome, Rome, Italy

Communicated by Shweta Agrawal

Received 6 March 2023 / Revised 24 February 2024 / Accepted 18 April 2024

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.

*An abridged version of this paper appears in the Proceedings of Advances in Cryptology-EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques [25].
"This paper was reviewed by Prabhanjan Ananth and David Wu."

Published online: 14 May 2024

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] x, whereas the secret key is associated to a predicate P, in such a way that the decryption process reveals the message if and only if the attribute x satisfies the predicate P (i.e., $P(x) = 1$). 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 dk_P, so long as $P(x^0) = P(x^1) = 0$ (otherwise it is easy to distinguish). The above security notion is also known as "weak" attribute-hiding which considers the secrecy of the attributes only in the case of a receiver not able to decrypt the ciphertext, i.e., the predicate is not satisfied.

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 et al. [30] give a construction of selectively secure PE (with unbounded collusions) for arbitrary predicates under the learning with errors (LWE) assumption. Moreover, under sub-exponential 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 toward 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.

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

Multi-key PE. In multi-key PE, we consider an ensemble of predicates $\mathcal{P} = \{P_v\}$ indexed by a value $v \in \mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$ which is uniquely represented as a sequence $v = (v_1, \ldots, v_n) \in \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$. 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 dk_{v_i} (using the corresponding master secret key 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 $P_v(x) = P_{v_1,\ldots,v_n}(x) = 1$.

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 $v_1, \ldots, v_n$ which correspond to predicates indexed by any value $v = (v_1, \ldots, v_n)$ such that $P_v(x^0) = P_v(x^1) = 0$. 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 $P_v(x^0) = P_v(x^1) = 1$, 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 $P_v(x) = P_{v_1}(x_1) \wedge \cdots \wedge P_{v_n}(x_n)$, where $x = (x_1, \ldots, x_n)$ and $v = (v_1, \ldots, v_n)$.

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 P with n inputs, i.e., predicates of the form $P(x_1, \ldots, x_n)$. A trusted authority produces encryption keys ek_i which are associated to the ith slot of an input for P; namely, given a (possibly secret)[2] encryption key 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 dk_P associated to an n-input predicate P, with the guarantee that the receiver can successfully decrypt $c_1, \ldots, c_n$, and thus obtain $m_1, \ldots, m_n$, so long as $P(x_1, \ldots, x_n) = 1$.

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 $(x_1^0, \ldots, x_n^0)$ and $(x_1^1, \ldots, x_n^1)$ and for any pair of sequences of messages $(m_1^0, \ldots, m_n^0)$ and $(m_1^1, \ldots, m_n^1)$, ciphertexts $c_1, \ldots, c_n$ corresponding to either $(x_1^0, m_1^0), \ldots, (x_n^0, m_n^0)$ or $(x_1^1, m_1^1), \ldots, (x_n^1, m_n^1)$ 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 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 ek_i (and thus corrupt a subset of the senders).

[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.

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 $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \cdots \wedge P_n(x_n)$ such that, for each $i \in [n]$, there exists a (public) wildcard input $x_i^\star$ for which $P_i(x_i^\star) = 1$ for every ith predicate P_i.[3] Our multi-input PE construction retains its security only in the setting of no corruptions (i.e., the encryption keys ek_i are kept secret) and no collusions (i.e., the adversary only knows a single decryption key dk_P for an adversarially chosen predicate 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.

Additional content of this manuscript. A preliminary version of this work appears in the Proceedings of EUROCRYPT 2023 [25]. Material not present in the Proceedings, but included in this manuscript, are (i) construction of multi-input PE in the setting of no corruptions (Construction 3 of Sect. 5.2); (ii) applications of our constructions (Sect. 6);

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

(*iii*) security proofs of our results including the ones contained in the Proceedings of EUROCRYPT 2023 [25] (Sect. 5).

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:

$$P(x_1, \ldots, x_n) = P_1(x_1) \wedge \cdots \wedge P_n(x_n). \tag{1}$$

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 Sects. 4 and 5.

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 P_v, within a family $\mathcal{P}$ of predicates indexed by values $v \in \mathcal{V}$, such that $P_v(x) = 1$. 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 $P_v(\cdot) = P_{v_1,\ldots,v_n}(\cdot)$, i.e., the decryption key associated to a particular predicate is decomposed into n decryption keys. Each decryption key 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 $P_{v_1,\ldots,v_n}(x) = 1$, where $P_{v_1,\ldots,v_n}(\cdot)$ 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 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 $P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 0$ (i.e., the adversary cannot decrypt the challenge ciphertext).[5]

As explained above, we focus on conjunctions of arbitrary predicates $P_{v_1,\ldots,v_n}(x) = P_{v_1,\ldots,v_n}(x_1, \ldots, x_n) = P_{v_1}(x_1) \wedge \cdots \wedge P_{v_n}(x_n)$ as defined in Eq. (1); hence, $x = (x_1, \ldots, x_n)$ and each dk_{v_i} identifies the ith 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 ith PE.[6] In a nutshell, our n-key PE scheme $\mathsf{kPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ works as follows:

Setup. The setup algorithm Setup simply executes Setup_i of each PE_i and outputs the master public key $\mathsf{mpk} = (\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n)$ and n master secret keys $(\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$.

Key Generation. To generate a decryption key $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, v_i)$ (representing the ith predicate $P_{v_i}(\cdot)$ of the conjunction), the authority can use the key generation algorithm of the ith PE, i.e., $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_i, P_{v_i})$.

Encryption. 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.,

$$c \leftarrow\!\!\$\ \mathsf{Enc}_n(\mathsf{mpk}_n, x_n, \mathsf{Enc}_{n-1}(\mathsf{mpk}_{n-1}, x_{n-1}, \ldots, \mathsf{Enc}_1(\mathsf{mpk}_1, x_1, y))).$$

Note that, for $n = \mathsf{poly}(\lambda)$, the final ciphertext will be of polynomial size since each underlying ith PE scheme has $\mathsf{poly}(\lambda)+|m_i|$ ciphertext expansion where $|m_i|$ is the message length supported by ith scheme. The final ciphertext of the n-key PE 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., $y = \mathbb{C}_c(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}) = \mathsf{Dec}_1(\mathsf{dk}_{v_1}, \ldots, \mathsf{Dec}_n(\mathsf{dk}_{v_n}, c))$.

Decryption. Finally, decryption is straightforward: the receiver simply executes $\widetilde{\mathbb{C}}$ using its n decryption keys $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$.

[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 $P_{v_1,\ldots,v_i,\ldots,v_n}$, the adversary can ask for an additional ith 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 $P_{v_1,\ldots,v'_i,\ldots,v_n} \neq P_{v_1,\ldots,v_i,\ldots,v_n}$.

[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 $s \leftarrow\!\!\$\ \leftarrow\!\!\$\ ^\lambda$.

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 A cannot hold a tuple of decryption keys $(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n})$ such that $P_{v_1,\ldots,v_n}(x^b) = P_{v_1,\ldots,v_n}(x_1^b, \ldots, x_n^b) = 1$, where $x^b = (x_1^b, \ldots, x_n^b)$ is the input chosen by A during the challenge phase, and b is the challenge bit. Since $P_{v_1,\ldots,v_n}(x_1^b, \ldots, x_n^b)$ is a conjunction of arbitrary predicates (see Eq. (1)), this implies that there exists an $i \in [n]$ such that $P_{v_i}(x_i^b) = 0$ for every ith decryption key dk_{v_i} obtained by A. We can leverage this observation together with the CPA security of PE_i to do a first hybrid in which the challenger computes the ith layer of the challenge ciphertext as $\mathsf{Enc}_i(\mathsf{mpk}_i, x_i^b, 0 \ldots 0)$. 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 $P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 1$ 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 msk. Each encryption key ek_i is given to the ith sender and allows the latter to handle the ith slot of a multi-input predicate. The ith party encrypts a message m_i under an input x_i by using its encryption key 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 dk_P associated to an n-input predicate P (recall that dk_P is generated by the authority via $\mathsf{KGen}(\mathsf{msk}, P)$) to execute $\mathsf{Dec}(\mathsf{dk}_P, c_1, \ldots, c_n)$. Intuitively, the decryption algorithm returns $(m_1, \ldots, m_n)$ when $P(x_1, \ldots, x_n) = 1$ where (m_i, x_i) are the message and the input associated to the ith 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 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 A. Naturally, this is subject to the usual validity condition, informally saying that 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. We recall

[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.

[8] Indeed, as we discuss in Sect. 4.3, 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.

that our multi-input constructions will support conjunctions of arbitrary predicates with wildcards (see Theorems 3 and 2 of Sect. 1.1).

Security in the secret-key setting. Here, no corruptions are allowed and thus the encryption keys are kept secrets. Hence, an adversary A playing the CPA-1-sided security game has adaptive oracle access to both the key generation 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 A to generate ciphertexts (associated to the ith 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 $(c_1^*, \ldots, c_n^*)$ with the ciphertexts obtained through the encryption oracles. This has a huge impact on the security of the a n-input PE scheme and on the validity condition that A must satisfy. For example, during the challenge phase, A could choose two vectors of messages $(m_1^0, \ldots, m_n^0)$ and $(m_1^1, \ldots, m_n^1)$ and two vectors of predicate inputs $(x_1^0, \ldots, x_n^0)$ and $(x_1^1, \ldots, x_n^1)$ such that for every predicate P (submitted to oracle $\mathsf{KGen}(m, \cdot)$) we have $P(x_1^0, \ldots, x_n^0) = P(x_1^1, \ldots, x_n^1) = 0$. Although the vector $(c_1^*, \ldots, c_n^*)$ cannot be directly decrypted, A could still be able to decrypt part of it by leveraging the encryption oracles. In more details, A could: (i) adversarially choose x_i' such that $P(x_1^0, \ldots, x_i', \ldots x_n^0) = 1$ and $P(x_1^1, \ldots, x_i', \ldots x_n^1) = 0$; (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 $(c_1^*, \ldots, c_i', \ldots, c_n^*)$. 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 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 dk_P corresponding to a predicate P, for every vector of ciphertexts obtained by interleaving the challenge ciphertext $(c_1^*, \ldots, c_n^*)$ with the ciphertexts generated through any of the n encryption oracles, we must have that P is not satisfied. This is formalized by the following condition: $\forall P \in \mathcal{Q}_{\mathsf{KGen}}, \forall j \in [n]$, $\forall i_1 \in [k_1 + 1], \ldots, \forall i_n \in [k_n + 1]$, it holds that

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

where $\mathcal{Q}_{\mathsf{KGen}}$ are the queries submitted to oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$, $(x_1^0, \ldots, x_n^0)$, $(x_1^1, \ldots, x_n^1)$ are the predicate inputs chosen by A during the challenge phase, and $\mathcal{Q}_i^b = \{x_i^{(1,b)}, \ldots, x_i^{(k_i,b)}, x_i^{(k_i+1,b)} = x_i^b\}$ 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. 4.2.

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 $P(x_1, \ldots, x_n)$ as defined in Eq. (1), where we view $(x_1, \ldots, x_n)$ as a single input chosen by the sender. In addition, the predicate must have a (efficiently computable) wildcard input $(x_1^\star, \ldots, x_n^\star)$ such that $x_i^\star$ satisfies every ith predicate of the conjunction, i.e., $P_i(x_i^\star) = 1$. As we will describe next, the $n-1$ subset of wildcards $(x_1^\star, \ldots, x_{i-1}^\star, x_{i+1}^\star, \ldots, x_n^\star)$ will permit the ith 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. We highlight that wildcards can be generically added to any single-input PE for arbitrary predicates. Let P the original predicate supported by the single-input PE scheme. Then, we can add a wildcard by translating P into a new predicate P' which admits a special (dummy) input $x^\star$ that always evaluate the predicate to 1, i.e.,

$$P'(x) = \begin{cases} 1 & \text{if } x = x^\star, \\ P(x) & \text{otherwise.} \end{cases}$$

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 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, our construction works as follows:

Encryption keys. The ith secret encryption key has the form $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$ where mpk is the master public key of the single-input PE, and k_i for $i \in [n]$ is a secret key for the SKE. (We also let $\mathsf{ek}_{n+1} = \mathsf{k}_1$.[9])

Encryption. In order to encrypt a message m_i under an input x_i, the ith sender samples a random lock y_i and encrypts (y_i, 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 dk_P for $P(x_1, \ldots, x_n)$, the circuit $\mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}$ acts as follows:

1. Set $\mathsf{k} = \mathsf{k}_{i+1}$ where k_{i+1} is the secret key hardcoded into the circuit (recall that secret keys are cyclically ordered, i.e., $\mathsf{k}_{n+1} = \mathsf{k}_1$).
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 k, i.e., $c_j^{(1)} = \mathsf{Dec}(\mathsf{k}, c_j^{(2)})$.

[9] In other words, the secret keys are cyclically ordered.

(a) Decrypt $c_j^{(1)}$ using dk_P in order to get (y_j, k_{j+1}). If $c_j^{(1)}$ decrypts correctly, 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 $(y_i, \mathsf{k}_{i+1}) = \mathsf{Dec}(\mathsf{dk}_P, \mathsf{Dec}(\mathsf{k}, c_i^{(2)}))$, 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).

Decryption. By following the computation (described above) of the obfuscated circuit, decryption is immediate. Upon input $(c_i)_{i\in[n]}$, the receiver computes $m_i = \widetilde{\mathbb{C}}_i(c_{i+1}^{(2)}, \ldots, c_n^{(2)}, c_1^{(2)}, \ldots, c_{i-1}^{(2)}, \mathsf{dk}_P)$ where $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$ and dk_P is the decryption key of the underlying single-input PE for a predicate $P(x_1, \ldots, x_n)$.

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 dk_P (0a), we are guaranteed that $P_i(x_i) = 1$ (recall that x_i is the input of the ith sender). In particular, the latter holds as, in any other slot, the ith sender has used the wildcards. By repeating this argument, we can conclude that $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \ldots \wedge P_n(x_n)$ is satisfied if the execution of each $\mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}$ goes as expected. The formal construction is described in Sect. 5.2.

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 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 P^* be the *only* predicate submitted to KGen by the adversary. Starting from A's validity condition, we infer that, for any choice of the challenge bit $b \in \leftarrow\!\$$, then attacker A must maintain one of the following two conditions:

(i) either $P_1^*(x_1^b) = \ldots = P_n^*(x_n^b) = 0$ (i.e., all the predicates of the conjunctions are false);
(ii) or (if at least one predicate P_i^* is satisfied, i.e., $P_i^*(x_i^b) = 1$) there exists $j \neq i$ such that, for every $x_j \in \mathcal{Q}_j^b$, it holds that $P_j^*(x_j) = 0$ 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)).[10]

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, k_{j+1})).

[10] 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 P^*.

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 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 $c^{(2)}_{j+1}$ by encrypting the all-zero string (instead of $c^{(1)}_{j+1}$ that, in turn, contains the lock y_{j+1} and the symmetric key 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.

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 dk_P is crucial in order to get the validity condition (ii), i.e., for every $b \in \leftarrow\!\!\$$ there exists a j such that for every predicate (submitted to $\mathsf{KGen}(\mathsf{msk}, \cdot)$) we have $P_j(x^b_j) = 0$. In fact, in the case of collusions, the adversary can ask for two decryption keys dk_P and $\mathsf{dk}_{P'}$ such that for every $b \in \leftarrow\!\!\$$:

$$P_1(x^b_1) = 0 \text{ and } P_2(x^b_2) = \ldots = P_n(x^b_n) = 1$$
$$P'_1(x^b_1) = 1 \text{ and } P'_2(x^b_2) = \ldots = P'_n(x^b_n) = 0.$$

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 KGen oracle of the single-input PE, making it invalid for the CPA-1-sided security of the single-input PE).[11]

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 $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \ldots \wedge P_n(x_n)$ with a *single* wildcard $(x^\star_1, \ldots, x^\star_n)$ into a secure multi-input PE for the same class of predicates $P(x_1, \ldots, x_n)$ *without the wildcard*. This can be done by requiring the senders not to encrypt the corresponding wildcard, i.e., for each $i \in [n]$, $\mathsf{Enc}(\mathsf{ek}_i, x^\star_i, m_i)$ outputs $\perp$ whenever $x_i = x^\star_i$. 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

[11] As we discuss in Sect. 5.4, our construction remains secure if we consider a weaker form of collusion in which the adversary can only obtain multiple decryption keys for predicates P such that there is a unique j *for all predicates* (submitted to KGen) that satisfies the validity condition (ii).

subset of the senders and leak their encryption keys ek_i. We model this by introducing an additional corruption oracle $\mathsf{Corr}(\cdot)$ that, upon input an index $i \in [n]$, returns ek_i. Note that, once obtained ek_i, the adversary 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 dk_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 through any of the (uncorrupted) encryption oracles and the ones that A may autonomously produce by using the leaked encryption keys (through oracle $\mathsf{Corr}(\cdot)$), we have that P is not satisfied. Hence, the validity condition is identical to that of the secret-key setting (see Eq. (2)), except that:

- If the ith encryption key ek_i has been corrupted/leaked, then $\mathcal{Q}_i^b$ of Eq. (2) corresponds to the ith predicate input space. This is because the adversary can produce a valid ciphertext on any input x_i.
- Else (i.e., the ith encryption key 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. 4.2 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. This will help us identifying the main properties that a multi-input scheme must satisfy in order to remain secure in case of corruptions. Suppose an adversary A has a single decryption key dk_P for $P(x_1, x_2) = P_1(x_1) \wedge P_2(x_2)$ 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 $P_1(x_1) = 0$ and $P_2(x_2) = 1$. Note that this ciphertext should not decrypt under dk_P, since the conjunction of P_1 and P_2 evaluates to 0. If A can obtain ek_2, then it can easily determine the message m_2 (and thus the bit b). Indeed, once 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 $P_1(x_1') = 1$ and $P_2(x_2') = 1$. Then, it can compute $\widetilde{c}_1^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{k}_1, \widetilde{c}_1^{(1)})$ and execute $\widetilde{\mathbb{C}}_2(\widetilde{c}_1^{(2)}, \mathsf{dk}_P)$ to get m_2. Note that by definition the execution of $\widetilde{\mathbb{C}}_2$ outputs the correct message, since $P_1(x_1') \wedge P_2(x_2) = 1$ and $\widetilde{c}_1^{(2)}$ contains the correct secret encryption key k_2, allowing the circuit to correctly end the computation. Also, note that this attack does not violate the validity condition. This is because $P_1(x_1) = 0$, and 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 $P(x_1, x_2') = P_1(x_1) \wedge P_2(x_2')$ unsatisfied for every choice of the input predicate x_2'.

In light of the above attack, we can identify the main properties that a multi-input PE scheme must satisfy to remain secure even in the presence of corruption:

1. Naturally, as for the secret-key setting, it is fundamental that the encrypted inputs and encrypted messages remain secret when one of the predicates P_i of the conjunction is not satisfied (see the proof strategy of our previous construction).

2. In combination with the above, we must guarantee that revealing one (or more) encryption key leaks no information about the encryption keys of other senders. This is fundamental otherwise a malicious sender may be able to impersonate and produce valid ciphertexts on behalf of others. This affects the security of the scheme since an adversary able to forge ciphertexts on behalf of an honest sender can violate the property described by the above Item 1 (i.e., the adversary can satisfy the ith predicate associated to the ith honest sender). We highlight that ensuring correctness while guaranteeing this property is challenging. For example, if the encryption key of the first sender "encodes" less information about the one of the second sender, then the harder will be the combination of their ciphertexts during decryption.

As demonstrated by the attack strategy described above, our secret-key multi-input PE scheme does not achieve the second property since an attacker can leak one encryption key which, in turn, allows it to produce a ciphertext on behalf of the honest sender (which allows for correct decryption in some scenarios).

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.

Encryption keys. We replace the SKE in our previous construction with a PKE, so that the encryption key ek_1 (resp. 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 ith sender. Indeed, note that only ek_i (the encryption key of the ith sender) contains the secret key sk_i). This is fundamental to deal with corruptions, i.e., corrupting the ith sender reveals no information about the secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_{i-1}, \mathsf{sk}_{i+1}, \ldots, \mathsf{sk}_n)$ of the other senders. Moreover, as we will next, sk_j will be required to generate valid ciphertexts for the jth slot of the scheme.

Encryption. 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 pk_1 and 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^{out} and message m_1, whereas the latter is obfuscated under the lock y_1^{in} and message sk_1. The ciphertext produced by the second sender, is identical, except that it uses sk_2 (instead of sk_1) and that $c_2^{(0)}$ is computed using the predicate input $(x_1^\star, x_2)$ (instead of $(x_1, x_2^\star)$).

[12] Recall that wildcards must be efficiently computable.

Decryption. The crux of our nesting technique comes from the definition of the circuits $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_i, c_i^{(2)}}$ which, in turn, defines the decryption algorithm of our construction (i.e., the nesting technique is fundamental to achieve correctness). More precisely, the outer circuit $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1, c_1^{(2)}}$ (i.e., the circuit that is given obfuscated to the receiver as part of the ciphertext c_1) 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 dk_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 $\widetilde{\mathbb{C}}_2^{\mathsf{in}}(\mathsf{sk}_1, \mathsf{dk}_P)$. At this point, $\widetilde{\mathbb{C}}_2^{\mathsf{in}}$ has everything it needs to check the satisfiability of $P_2(\cdot)$. 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 $(y_2^{\mathsf{in}}, y_2^{\mathsf{out}}) = \mathsf{Dec}(\mathsf{dk}_P, c_2^{(0)})$—observe that the decryption succeeds if $P_2(x_2) = 1$—and returns y_2^{in}. By correctness of lockable obfuscation, if the computation of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(2)}}(\mathsf{sk}_1, \mathsf{dk}_P)$ goes as intended, then $\widetilde{\mathbb{C}}_2^{\mathsf{in}}(\mathsf{sk}_1, \mathsf{dk}_P)$ will output sk_2 (the message attached to the obfuscation). Once obtained 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 $P_1(\cdot)$ except that, if the PE ciphertext $c_1^{(0)}$ decrypts correctly, it returns y_1^{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.

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 sk_i remains unknown to the adversary if $P_i(\cdot) = 0$ (unless the adversary invokes the oracle $\mathsf{Corr}(i)$). This is reflected by the proof technique that is sketched below.

Let dk_{P^*} be the decryption key obtained by A for the predicate $P^*(\cdot,\cdot) = P_1^*(\cdot) \wedge P_2^*(\cdot)$ (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 $b \in \leftarrow\!\$$ we have:

(i) either $P_1^*(x_1^b) = P_2^*(x_2^b) = 0$;
(ii) or (i.e., there exists an $i \in [2]$ such that predicate P_i is satisfied) $j \notin \mathcal{Q}_{\mathsf{Corr}}$ such that $j \neq i$ and, for every $x_j \in \mathcal{Q}_j^b$, $P_j^*(x_j) = 0$ (recall that $x_j^b \in \mathcal{Q}_j^b$). Observe that this second condition holds because of the following:

- If there is $x_j \in \mathcal{Q}_j^b$ such that $P_j^*(x_j) = 1$, A can use the corresponding ciphertext to decrypt the ith part of the challenge ciphertext since $P_i^*(x_i^b) = 1$.
- If $j \in \mathcal{Q}_{\mathsf{Corr}}$, A can simply use ek_j to encrypt a random message under the wildcard $x_j^\star$ (that always exists by design of our construction) and, again, decrypt the ith 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 ek_j gets

leaked then a malicious adversary can always encrypt itself the jth wildcards $x_j^\star$, satisfying the jth predicate 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 $(y_1^{\mathsf{in}}, y_1^{\mathsf{out}})$ and $(y_2^{\mathsf{in}}, y_2^{\mathsf{out}})$ (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 $(\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1,c_1^{(2)}}, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1,c_1^{(2)}})$, $(\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_2,c_2^{(2)}}, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2,c_2^{(2)}})$, 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 $c_j^{(0)}$ by encrypting the all-zero string (instead of $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}})$). 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 sk_j of the uncorrupted jth 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 $(y_i^{\mathsf{in}}, y_i^{\mathsf{out}})$ chosen by the ith sender (recall $i \neq j$). In more details, we do another hybrid in which the jth PKE layer $c_i^{(j)}$ of the challenge ciphertext is an encryption of zeroes (instead of $c_i^{(j-1)}$ that, in turn, encrypts the locks $(y_i^{\mathsf{in}}, y_i^{\mathsf{out}})$). 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 ith 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 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 ek_i is of the form $(\mathsf{mpk}, \mathsf{sk}_i, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$.
- To compute the ith 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 dk_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 $i \in [n]\backslash\{1\}$, 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 dk_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 ($\mathsf{sk}_1, \perp, \ldots, \perp$), dk_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 ($\mathsf{sk}_1, \mathsf{sk}_2, \perp, \ldots, \perp$), dk_P, and the remaining obfuscations ($\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_4,c_4^{(n)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$).
- The above process is repeated until $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$ is executed upon input ($\mathsf{sk}_1, \ldots, \mathsf{sk}_{n-1}, \perp$) and dk_P. At this point, all the secret keys are known (observe that sk_n is hard-coded). From $c_n^{(n)}$, we can remove the n PKE layers, decrypt the PE ciphertext and, in turn, return y_n^{in} if the PE ciphertext decrypts correctly (i.e., $P_n(\cdot)$ is satisfied).
- Once $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$ terminates, the secret key sk_n is released and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_{n-1}^{(n)}}$ performs the computation required to check if $P_{n-1}(\cdot)$ is satisfied. Indeed, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_{n-1}^{(n)}}$ has been executed on input ($\mathsf{sk}_1, \ldots, \mathsf{sk}_{n-2}, \perp, \perp$), it has sk_{n-1} harcoded, and the execution of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_n,c_n^{(n)}}$ has released 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 sk_n. This is because the latter circuit receives as input ($\mathsf{sk}_1, \ldots, \mathsf{sk}_{n-3}, \perp, \perp, \perp$), it has sk_{n-2} hardcoded, and the circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_n^{(n)}}$ has returned 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 ($\mathsf{sk}_1, \ldots, \mathsf{sk}_{n-1}, \perp$) in order to get 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. 5.3.

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.

Matchmaking Encryption. Matchmaking encryption (ME) [10,11] allows a sender to publicly encrypt a message m under some attributes σ and a policy $\mathbb{R}$. On the other hand, the receiver can use the decryption keys dk_ρ and $\mathsf{dk}_\mathbb{S}$ (encoding the receiver's attributes and policy, respectively) to decrypt the message (i.e., $\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c) = m$) if there is a mutual match $\mathbb{S}(\sigma) = 1 \wedge \mathbb{R}(\rho) = 1$. The main security guarantee of ME is defined by the following two properties:

- In case of a mismatch, nothing is leaked except the fact that a match did not occur.
- Additionally, in case of a match, nothing is leaked except for the message and the fact that a match occurred.

These properties are reminiscent to CPA-2-sided security of PE. Multi-key PE is a direct generalization of ME: 2-key PE for conjunctions $P_{v_1,v_2}(\cdot,\cdot) = P_{v_1}(\cdot) \wedge P_{v_2}(\cdot)$ (i.e., the class of predicates studied in this work) implies ME for arbitrary policies. In a nutshell, the construction works as follows. To encrypt a message m under the sender's attributes σ and the sender's policy $\mathbb{R}$, the ME encryption algorithm corresponds to the public-key encryption algorithm of the 2-key PE scheme, i.e., $c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, (x_1, x_2), m)$ where $x_1 = \sigma$ and $x_2 = \mathbb{R}$. Analogously, the ME decryption keys dk_ρ and $\mathsf{dk}_\mathbb{S}$ correspond to the decryption keys dk_{v_2} and dk_{v_1} of the 2-key PE scheme where $v_1 = \mathbb{S}$ and $v_2 = \rho$. By setting $P_{v_1,v_2}(x_1, x_2) = P_{\mathbb{S},\rho}(\sigma, \mathbb{R}) = P_\sigma(\mathbb{S}) \wedge P_\mathbb{R}(\rho) = \mathbb{S}(\sigma) \wedge \mathbb{R}(\rho)$, we obtain the desired ME functionality during decryption. The security analysis is intuitive: if the 2-key PE is CPA-1-sided secure, then the ME scheme is secure only in case of mismatch. In addition, if the 2-key PE is CPA-2-sided secure, then the ME security holds also in case of a match. Hence, as a corollary of our results, we achieve the weaker notion of CPA-1-sided secure (i.e., mismatch) ME supporting arbitrary policies and unbounded collusions from sub-exponential LWE. We provide more details in Sect. 6.1.

The seminal works of ME [10,11] propose ME as a tool for anonymous communication with bilateral authentication. The anonymity level guaranteed by the scheme

depends on the notion of security. ME with CPA-2-sided security (as originally proposed by [10,11]) guarantees the anonymity of users (e.g., users' attributes and policies) independently from the outcome of the bilateral matching (i.e., match and mismatch). In the case of CPA-1-sided secure ME (as the one proposed in this work), anonymity is guaranteed only in case of a mismatch, i.e., unauthorized parties infer no information about the identity of the sender. Thus, our notion of CPA-1-secure ME can be used in scenarios in which the sender's identity can be disclosed to authorized receivers (e.g., health-care scenarios where a bilateral matching between patients and doctors is performed).

Previous works construct CPA-2-sided secure ME with unbounded collusions for either very restricted policies (i.e., for identity matching) using bilinear maps [20,26] (and ROM [10]), or for arbitrary policies from much stronger assumptions such as 2-input FE with one secret key and one public key (this notion of 2-input FE implies iO) [10,11].

For completeness (see Sects. 4.1, 4.3), we highlight that we can build n-key PE from $(n+1)$-input PE supporting arbitrary predicates and tolerating 1 corruption (this is required to implement the public-key encryption algorithm of n-key PE). As a consequence, multi-input PE implies ME as well. However, recall that our multi-input PE constructions do not support arbitrary predicates but only conjunctions of arbitrary predicates with wildcards.

Non-interactive MPC. Non-interactive 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 crs and encryption keys ek_i) that can be later used by the parties to perform function evaluation. Security of NI-MPC can be formulated in two different settings, named *non-reusable* and *reusable* NI-MPC. The former retains security only if the setup is executed after every round. The latter retains security even if parties evaluate f on different inputs using the same setup (full-fledged reusability makes use of session identifiers in order to avoid that an adversary can interleave messages from different rounds [34]). Both non-reusable and reusable NI-MPC provide the same security guarantee, formalized using an indistinguishability-based definition: an adversary 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 ek_i of a corrupted party) yields the same function's evaluation.[13]

As mentioned by several works [14,29,32,33], NI-MPC achieving indistinguishability-based security implies iO even in very restricted settings. In particular, a non-reusable 1-robust (i.e., one malicious party) NI-MPC for two parties implies iO. Intuitively, by fixing the NI-MPC function to $f(\mathbb{C}, x) = \mathbb{C}(x)$, we can obfuscate a circuit by simply setting the input of the first (honest) party to $\mathbb{C}$, compute $c_1 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_1, \mathbb{C})$, and outputting $\widetilde{\mathbb{C}} = (\mathsf{crs}, c_1, \mathsf{ek}_2)$ where $\mathsf{ek}_1, \mathsf{ek}_2$ are the key material required to encode 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].

inputs of the NI-MPC (note that 1-robustness is necessary since we reveal ek_2). To evaluate the obfuscated circuit, the evaluator only needs to compute $c_2 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_2, x)$ and evaluate the NI-MPC function f that will yield $\mathbb{C}(x)$. The security of this iO obfuscator follows from the security of NI-MPC since the residual functions $f(\mathbb{C}_0, \cdot)$ and $f(\mathbb{C}_1, \cdot)$ are identical, as $\mathbb{C}_0(x) = \mathbb{C}_1(x)$ for every input x. Additionally, reusable, 0-robust (i.e., no malicious parties) NI-MPC for $n = \mathsf{poly}(\lambda)$ parties implies iO. In this case, iO can be built using a similar construction to that of iO from secret-key multi-input functional encryption (FE) [29].

Due to the similarities between multi-input PE and multi-input FE, we observe that multi-input PE is enough to construct NI-MPC for *all-or-nothing* functions defined over the predicates supported by the multi-input PE scheme. In more details, by leveraging our CPA-1-sided n-input PE (for $n = O(1)$) secure under $n-1$ corruptions and without collusions, we can build an $(n-1)$-robust NI-MPC for a constant number of parties for the following class of functions:

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

where $P(x_1, \ldots, x_n)$ is a conjunctions of arbitrary independent predicates (with wildcards) as defined in Eq. (1). 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 the usual indistinguishability-based security only if f_P outputs $\perp$ (i.e., $P(\cdot)$ is not satisfied) for any combination of the honest messages and the ones the adversary can maliciously compute using the encryption key ek_i of a corrupted party.

The construction is intuitive. At setup, simply publish $\mathsf{crs} = \mathsf{dk}_P$ and distribute ek_i to the ith party where $(\mathsf{msk}, \mathsf{ek}_1, \ldots \mathsf{ek}_n) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and $\mathsf{dk}_P \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, P)$. During evaluation, each party can send the message $c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$ and compute $\mathsf{Dec}(\mathsf{dk}_P, c_1, \ldots, c_n)$ to evaluate the function $f_P((x_1, m_1), \ldots, (x_n, m_n))$. The CPA-1-sided reusable security of k-robust NI-MPC for f_P follows readily from CPA-1-sided security of n-input PE under k corruptions and without collusions.

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 P of the function f_P is a conjunctions of arbitrary predicates (i.e., $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \cdots \wedge P_n(x_n)$) with wildcards under the LWE assumption.

An example of an application of (CPA-1-sided reusable) NI-MPC is one-round voting protocols: We imagine the scenario where a committee consisting of n parties wants to approve a certain law. They can use NI-MPC to encode their set of constraints as their input x_i. The law is then approved if $P_1(x_1) \wedge \cdots \wedge P_2(x_n) = 1$, where P_i is a (public) policy that checks if the constraint imposed by x_i is satisfied by the law. Importantly, the protocol is completely non-interactive, and therefore the parties can just send their messages and go offline, without the need to wait for everyone to respond. In terms of security, the law is approved only if all policies are satisfied and otherwise the preference of each party is kept hidden. For instance, a hypothetical party that blocked the law would

remain anonymous. We provide the formal definition of CPA-1-sided reusability and the construction of NI-MPC from multi-input PE in Sect. 6.2.

We emphasize that, nonetheless CPA-1-sided reusability is a weakening of the standard reusability definition, our flavor of reusability is 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. The above observation motivates our research question of building such a notion of NI-MPC for restricted functionalities. Considering restricted functionalities, such as conjunction of arbitrary predicates, permits us to construct NI-MPC from LWE that is, at the time of this writing, a computational assumption not sufficient for constructing null iO and witness encryption.

1.4. *Relation with Witness Encryption*

In the following we recall the notion of witness encryption (WE) [27], and we discuss its relation with both multi-input and multi-key schemes. We anticipate that such relations do not require CPA-1-sided and CPA-2-sided security. Hence, the following discussion will focus on multi-input and multi-key ABE schemes, i.e., predicate inputs can be public.

A WE scheme for a relation $\mathcal{R}$, defined over a language $\mathcal{L}$, allows a sender to encrypt a message m using a statement x. A receiver, holding a witness w, can decrypt the message m if $(x, w) \in \mathcal{R}$. As for security, WE guarantees that the message remains hidden whenever $x \notin \mathcal{L}$, i.e., the corresponding ciphertext cannot be decrypted. WE has several disrupting applications such as encrypting messages that can be decrypted in future (i.e., whenever w will be known). Moreover, WE does not require setup and is fully non-interactive.

As shown by Brakerski et al. [19], an n-input ABE (i.e., predicate inputs can be public) for arbitrary predicates (or any predicate that "match" the desired NP relation), secure in the secret-key setting and without collusions, implies WE for NP and n-size witnesses. The construction is reminiscent to the one of iO from secret-key multi-input functional encryption [29] (see also Sect. 1.3). Unfortunately, we cannot use here our n-input scheme since it only supports conjunctions of arbitrary predicates (see Eq. (1)). Currently, it is not known how to build n-input ABE (and thus PE), with $n > 2$, for arbitrary predicates without iO (the only known construction is for $n = 2$ and it is due to the work of Agrawal et al. [8]. See Sect. 2 for a detailed discussion.

Also, we stress that multi-key ABE (i.e., a multi-key scheme where predicate inputs can be public) for arbitrary predicates implies WE. The construction is similar to that of Brakerski et al. [19], for obtaining WE from multi-input ABE. The only difference is that we substitute the multiple inputs with the multiple decryption keys of multi-key ABE. For completeness, we describe the construction below. Let $P_{v_1,\ldots,v_n}(x) = 1$ if and only if $(x, w) \in \mathcal{R}$, where $w = v_1||\ldots||v_n$ defines the class of predicates supported by the multi-key ABE. To encrypt a message m under a statement $x \in \mathcal{L}$, the sender computes $(\mathsf{mpk}, \mathsf{msk}_1, \ldots, \mathsf{msk}_n) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and sends to the receiver $(c, (\mathsf{dk}_{v_i}, \mathsf{dk}_{\overline{v_i}})_{i\in[n]})$ where $c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, x, m)$ and $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, 1)$ (resp.

$\mathsf{dk}_{\overline{v_i}} \leftarrow\!\!\$\, \mathsf{KGen}(\mathsf{msk}_i, 0))$ for $i \in [n]$. To decrypt the ciphertext under a witness $w = v_1||\ldots||v_n$, the receiver simply executes $\mathsf{Dec}(\mathsf{dk}'_{v_1}, \ldots, \mathsf{dk}'_{v_n}, c)$ where $\mathsf{dk}'_{v_i} = \mathsf{dk}_{v_i}$ if $v_i = 1$, and $\mathsf{dk}'_{v_i} = \mathsf{dk}_{\overline{v_i}}$ if $v_i = 0$.[14] Similarly to the case of multi-input, our multi-key construction fails to imply WE since it does not support arbitrary predicates (we stress once again that CPA-1-sided and CPA-2-sided security are not required).

It may seem that arbitrary predicates are a necessary condition in order to build WE from multi-input schemes. However, we highlight that this is not necessarily the case if we consider security under corruptions. In particular, a 2-input scheme for conjunctions under 1 corruption and no collusions, implies WE for any relation. This can be accomplished by considering the predicate $P_{x,\mathcal{R}}(\cdot,\cdot) = P_1(\cdot) \wedge P_{x,\mathcal{R}}(\cdot)$ such that $P_1(x_1^\star) = 1$ (for some wildcard $x_1^\star$) and $P_{x,\mathcal{R}}(w) = 1$ if and only if $(x, w) \in \mathcal{R}$. Intuitively, to encrypt m using a statement x, the sender can simply output $(c_1, \mathsf{ek}_2, \mathsf{dk}_{P_{x,\mathcal{R}}})$ such that $c_1 \leftarrow\!\!\$\, \mathsf{Enc}(\mathsf{ek}_1, x_1^\star, m)$, $\mathsf{dk}_{P_{x,\mathcal{R}}} \leftarrow\!\!\$\, \mathsf{KGen}(m, P_{x,\mathcal{R}})$, and $(\mathsf{msk}, \mathsf{ek}_1, \mathsf{ek}_2) \leftarrow\!\!\$\, \mathsf{Setup}(1^\lambda)$. Then, the receiver uses w to retrieve m by computing $\mathsf{Dec}(\mathsf{dk}_{x,\mathcal{R}}, c_1, \mathsf{Enc}(\mathsf{ek}_2, w))$.[15] Here, it is crucial that the underlying 2-input scheme can handle corruptions, since the latter allows the sender to disclose ek_2 to the (possibly malicious) receiver and give him the opportunity to try different witnesses.

Unfortunately, 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.[16]

Given the above discussion, we identify two plausible approaches that could lead to a construction of WE from standard assumptions:

- Enlarging the class of predicates of our secret-key n-input or n-key constructions: From conjunction of arbitrary predicates (see Eq. (1)) to arbitrary predicates (or any restricted class of predicates that permits to implement a specific non-trivial WE relation $\mathcal{R}$).
- Supporting conjunctions of arbitrary predicates (without wildcards) in the setting of 2-input 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]. The multi-input and multi-key settings have also been considered in the context of fully-homomorphic encryption [23,40,41].

[14] Observe that the same construction works if we start from a multi-key PE whose encryption algorithm is secret-key, i.e., the mpk (required to execute Enc) is replaced with an encryption key ek that is kept secret.

[15] A similar construction can be used to build iO from 2-input FE with security under 1 corruption and no collusions.

[16] If wildcards exist, a malicious receiver can always decrypt the message by evaluating the predicate over the wildcards.

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 work of Agrawal et al. [8] that proposes two constructions of secret-key (i.e., no corruptions) 2-input key-policy ABE for 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 P from lattices, and the second is a 3-input ABE for 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

3.1. *Notation*

We use the notation $[n] = \{1, 2, \ldots, n\}$. Capital bold-face letters (such as $\mathbf{X}$) are used to denote random variables, small letters (such as x) to denote concrete values, calligraphic letters (such as $\mathcal{X}$) to denote sets, serif letters (such as A) to denote algorithms, and bold typeface letters (such as $\mathbb{C}$) to denote circuits. All of our algorithms are modeled as (possibly interactive) Turing machines; if algorithm A has oracle access to some oracle O, we often implicitly write $\mathcal{Q}_{\mathsf{O}}$ for the set of queries asked by A to O.

For a string $x \in \leftarrow\!\$\, ^*$, we let $|x|$ be its length; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the cardinality of $\mathcal{X}$. When x is chosen uniformly in $\mathcal{X}$, we write $x \leftarrow\!\$\, \mathcal{X}$. If A is an algorithm, we write $y \leftarrow\!\$\, \mathsf{A}(x)$ to denote a run of A on input x and output y; if A is randomized, y is a random variable and $\mathsf{A}(x; r)$ denotes a run of A on input x and (uniform) randomness r. An algorithm A is *probabilistic polynomial-time* (PPT) if A is randomized and for any input $x, r \in \leftarrow\!\$\, ^*$ the computation of $\mathsf{A}(x; r)$ terminates in a polynomial number of steps (in the input size). We write $\mathbb{C}(x) = y$ to denote the evaluation of the circuit $\mathbb{C}$ on input x and output y.

Let $\mathbf{G}$ be an experiment defining the security of a cryptographic primitive Π and $\mathbf{E}$ be an event. We write $\mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}(\lambda) = 1 | \mathbf{E}\right]$ (i.e., the outcome of experiment $\mathbf{G}_{\Pi,\mathsf{A}}(\lambda)$ conditioned to the event $\mathbf{E}$) to denote the advantage of an adversary A in winning the experiment $\mathbf{G}_{\Pi,\mathsf{A}}(\lambda)$ (i.e., $\mathbf{G}_{\Pi,\mathsf{A}}(\lambda) = 1$) when the event $\mathbf{E}$ holds.[17]

Negligible functions. Throughout the paper, we denote by $\lambda \in \mathbb{N}$ the security parameter and we implicitly assume that every algorithm takes as input the security parameter. A function $\nu(\cdot)$ is called negligible in the security parameter $\lambda \in \mathbb{N}$ if it vanishes faster than the inverse of any polynomial in λ, i.e. $\nu(\lambda) \in O(1/p(\lambda))$ for all positive polynomials $p(\lambda)$. We sometimes write $\mathsf{negl}(\lambda)$ (resp. $\mathsf{poly}(\lambda)$) to denote an unspecified negligible function (resp. polynomial function) in the security parameter.

3.2. *Lockable Obfuscation*

A lockable obfuscator [31,48] permits to obfuscate a circuit $\mathbb{C}$ together with a "lock" y and a message m. As a result, the obfuscator will output an obfuscated circuit $\widetilde{\mathbb{C}}$ that will behave as follows:

$$\widetilde{\mathbb{C}}(x) = \begin{cases} m & \text{if } \mathbb{C}(x) = y \\ \perp & \text{otherwise.} \end{cases}$$

More formally, let $n(\cdot)$, $s(\cdot)$, $d(\cdot)$ be polynomials, and $\mathcal{C}_{n,s,d}(\lambda)$ be the family of circuits of depth $d(\lambda)$ with input size $n(\lambda)$ and output size $s(\lambda)$. A lockable obfuscator for the circuit family $\mathcal{C}_{n,s,d}(\lambda)$ and message space $\mathcal{M}$ is composed of the following polynomial-time algorithms:

[17] This is equivalent to saying that the output of $\mathbf{G}_{\Pi,\mathsf{A}}(\lambda)$ is set to 0 when $\mathbf{E}$ does not hold.

$\mathbf{G}^{\mathsf{lock\text{-}sim}}_{\Pi,\mathsf{A},\mathsf{S}}(\lambda)$

$(\mathbb{C}, m, \alpha) \leftarrow\!\$\ \mathsf{A}_0(1^\lambda)$

$b \leftarrow\!\$\ \{0,1\}, y \leftarrow\!\$\ \{0,1\}^{s(\lambda)}$

$\widetilde{\mathbb{C}}_0 \leftarrow\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}, y, m), \widetilde{\mathbb{C}}_1 \leftarrow\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{C}|}, 1^{|m|})$

$b' \leftarrow\!\$\ \mathsf{A}_1(1^\lambda, \widetilde{\mathbb{C}}_b, \alpha)$

If $(b' = b)$**: return** 1

Else: return 0

Fig. 1. Game defining security of lockable obfuscation.

$\mathsf{Obf}(1^\lambda, \mathbb{C}, y, m)$: Upon input the security parameter 1^λ, a circuit $\mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$, a lock $y \in \leftarrow\!\$^{s(\lambda)}$, and a message $m \in \mathcal{M}$, the randomized lockable obfuscator algorithm outputs a circuit $\widetilde{\mathbb{C}}$.

$\mathsf{Eval}(\widetilde{\mathbb{C}}, x)$: Upon input an obfuscated circuit $\widetilde{\mathbb{C}}$ and an input $x \in \leftarrow\!\$^{n(\lambda)}$, the deterministic evaluation algorithm outputs a message $m \in \mathcal{M} \cup \{\perp\}$.

Definition 1. (*Semi-statistical correctness of lockable obfuscation* [31]). A lockable obfuscator $\Pi = (\mathsf{Obf}, \mathsf{Eval})$ for the circuit family $\mathcal{C}_{n,s,d}(\lambda)$ and message space $\mathcal{M}$ satisfies *semi-statistical correctness* if:

1. $\forall \lambda \in \mathbb{N}, \forall x \in \leftarrow\!\$^{n(\lambda)}, m \in \mathcal{M}, \forall \mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$ such that $\mathbb{C}(x) = y$, we have
$$\mathbb{P}\left[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, \mathbb{C}, y, m), x) = m\right] = 1.$$
2. $\forall \lambda \in \mathbb{N}, \forall x \in \leftarrow\!\$^{n(\lambda)}, \forall m \in \mathcal{M}, \forall \mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$ such that $\mathbb{C}(x) \neq y$, we have
$$\mathbb{P}\left[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, \mathbb{C}, y, m), x) = m\right] \leq \mathsf{negl}(\lambda).$$

As for security, lockable obfuscation must hide any information about the circuit $\mathbb{C}$, the message m and the lock y when the lock is randomly chosen. This is defined by requiring that there exists a simulator S that simulates the obfuscated circuit $\widetilde{\mathbb{C}}$.

Definition 2. (*Security of lockable obfuscation*). A lockable obfuscator $\Pi = (\mathsf{Obf}, \mathsf{Eval})$ for the circuit family $\mathcal{C}_{n,s,d}(\lambda)$ and message space $\mathcal{M}$ is secure if there exists a PPT simulator S such that for every PPT adversary $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ we have:
$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{lock\text{-}sim}}_{\Pi,\mathsf{A},\mathsf{S}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$
where $\mathbf{G}^{\mathsf{lock\text{-}sim}}_{\Pi,\mathsf{A},\mathsf{S}}(\lambda)$ is depicted in Fig. 1.

Remark 1. The definitions above are taken from [31]. Wichs and Zirdelis [48] proposed a slightly more general notion of obfuscation for multi-bit compute-and-compare circuits in which the lock is only required to be unpredictable. They also give an obfuscator for multi-bit compute-and-compare circuits from the LWE assumption.

$\mathbf{G}^{\mathsf{CPA\text{-}ske}}_{\Pi,\mathsf{A}}(\lambda)$

$\mathsf{k} \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$

$(m^0, m^1, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{Enc}(\mathsf{k},\cdot)}(1^\lambda)$

$b \leftarrow\!\!\$\ \{0,1\}, c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{k}, m^b)$

$b' \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{Enc}(\mathsf{k},\cdot)}(1^\lambda, c, \alpha)$

If $(b' = b)$**: return** 1

Else: return 0

$\mathbf{G}^{\mathsf{CPA\text{-}pke}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$

$(m^0, m^1, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0(1^\lambda, \mathsf{pk})$

$b \leftarrow\!\!\$\ \{0,1\}, c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, m^b)$

$b' \leftarrow\!\!\$\ \mathsf{A}_1(1^\lambda, c, \alpha)$

If $(b' = b)$**: return** 1

Else: return 0

Fig. 2. Game defining CPA security of SKE and PKE.

3.3. *Symmetric and Public Key Encryption*

3.3.1. *Symmetric Key Encryption*

A symmetric-key encryption (SKE) scheme with message space $\mathcal{M}$ is composed of the following polynomial-time algorithms:

$\mathsf{KGen}(1^\lambda)$: The randomized key generator takes as input the security parameter 1^λ and outputs a symmetric key k.

$\mathsf{Enc}(\mathsf{k}, m)$: The randomized encryption algorithm takes as input a symmetric key k and a message $m \in \mathcal{M}$, and outputs a ciphertext c.

$\mathsf{Dec}(\mathsf{k}, c)$: The deterministic decryption algorithm takes as input a symmetric key k and a ciphertext c, and outputs a message m.

We require a SKE to be correct and secure against chosen-plaintext attacks (CPA).

Definition 3. (*Correctness of SKE*). A SKE Π with message space $\mathcal{M}$ is correct if $\forall\lambda \in \mathbb{N}, \forall m \in \mathcal{M}$, we have

$$\mathbb{P}\left[\mathsf{Dec}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, m)) = m\right] \geq 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{k} \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$. The above probability is taken over the random coins of KGen and Enc.

Definition 4. (*CPA security of SKE*). We say that a SKE Π is CPA secure if for all PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}^{\mathsf{CPA\text{-}ske}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 2.

3.3.2. *Public Key Encryption*

A public-key encryption (PKE) scheme with message space $\mathcal{M}$ is composed of the following polynomial-time algorithms:

$\mathsf{KGen}(1^\lambda)$: The randomized key generator takes as input the security parameter 1^λ and outputs a public and a secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m)$: The randomized encryption algorithm takes as input a public key pk and a message $m \in \mathcal{M}$ and outputs a ciphertext c.

$\mathsf{Dec}(\mathsf{sk}, c)$: The deterministic decryption algorithm takes as input a secret key sk and a ciphertext c and outputs a message m.

We consider the usual definition of correctness and CPA security of PKE.

Definition 5. (*Correctness of PKE*). A PKE Π with message space $\mathcal{M}$ is correct if $\forall \lambda \in \mathbb{N}$, $\forall(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{KGen}(1^\lambda)$, $\forall m \in \mathcal{M}$, we have

$$\mathbb{P}\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m\right] \geq 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$. The above probability is taken over the random coins of KGen and Enc.

Definition 6. (*CPA security of PKE*). We say that a SKE Π is CPA secure if for all PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}^{\mathsf{CPA\text{-}pke}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 2.

3.4. *Predicate Encryption*

In PE, a trusted authority generates a decryption key for the receiver associated to an arbitrary predicate of his choice. The receiver is able to decrypt a ciphertext if and only if the predicate P (corresponding to its decryption key) is satisfied when evaluated with the predicate input x used for encrypting the plaintext, i.e. $P(x) = 1$. Formally, a PE with message space $\mathcal{M}$, input space $\mathcal{X}$, and predicate space $\mathcal{P}$, is composed of the following polynomial-time algorithms:

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

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

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

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

Fig. 3. Game defining CPA, CPA-1-sided, and CPA-2-sided security of PE.

$\mathsf{Dec}(\mathsf{dk}_P, c)$: The deterministic decryption algorithm takes as input a secret decryption key dk_P for predicate $P \in \mathcal{P}$ and a ciphertext c. The algorithm outputs either a message m or an error $\perp$.

Correctness of PE states that the receiver obtains the message with overwhelming probability if $P(x) = 1$. On the other hand, if $P(x) = 0$, the decryption outputs $\perp$ with overwhelming probability.

Definition 7. (*Correctness of PE*). A PE with message space $\mathcal{M}$, input space $\mathcal{X}$, predicate space $\mathcal{P}$, is correct if $\forall\lambda \in \mathbb{N}$, $\forall m \in \mathcal{M}$, $\forall x \in \mathcal{X}$, $\forall P \in \mathcal{P}$, the following two conditions hold:

1. If $P(x) = 1$, then $\mathbb{P}\left[\mathsf{Dec}(\mathsf{dk}_P, \mathsf{Enc}(\mathsf{mpk}, x, m)) = m\right] \geq 1 - \mathsf{negl}(\lambda)$ where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and $\mathsf{dk}_P \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, P)$.
2. If $P(x) = 0$, then $\mathbb{P}\left[\mathsf{Dec}(\mathsf{dk}_P, \mathsf{Enc}(\mathsf{mpk}, x, m)) = \perp\right] \geq 1 - \mathsf{negl}(\lambda)$ where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and $\mathsf{dk}_P \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, P)$.

The above two probabilities are taken over the random coins of Setup, KGen and Enc.

Security of PE comes in different flavors. The standard CPA security requires the adversary to distinguish between the encryption of two messages for the same predicate input. More formally, the adversary is allowed to perform a polynomial number of queries to the key generation oracle. Then, the adversary chooses two messages m^0 and m^1 and an input x, and wins the CPA security game if it can distinguish between an encryption of $\mathsf{Enc}(\mathsf{mpk}, x, m^0)$ and $\mathsf{Enc}(\mathsf{mpk}, x, m^1)$ with non-negligible probability (a PE scheme that satisfies CPA security is also called attribute-based encryption (ABE)).

Definition 8. (*CPA security of PE*). We say that a PE Π is CPA secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}^{\text{CPA-PE}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 3. Adversary A is called valid if $\forall P \in \mathcal{Q}_{\mathsf{KGen}}$ it holds that $P(x) = 0$.

We also consider two stronger definitions of security, namely CPA-1-sided and CPA-2-sided security, guaranteeing also the secrecy of the predicate input used during the

encryption of a message. In this security games, the adversary is allowed to choose two different inputs x^0 and x^1 and the usual messages m^0 and m^1. CPA-1-sided security guarantees the privacy of the input only when the predicates for which the adversary knows a decryption key (i.e. the ones he received from the key generation oracle) are not satisfied, i.e. the receiver cannot decrypt the message. On the other hand, CPA-2-sided security considers the same property also when the predicate is satisfied, i.e., the receiver can decrypt the challenge ciphertexts.

Definition 9. (*CPA-1-sided and CPA-2-sided security of PE*). Let $t \in [2]$. We say that a PE Π is CPA-t-sided secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{CPA}\text{-}t\text{-}\mathsf{PE}}(\lambda)$ is depicted in Fig. 3. Adversary A is called valid if $\forall P \in \mathcal{Q}_{\mathsf{KGen}}$,

$$\textbf{Case } t = 1 : P(x^0) = P(x^1) = 0.$$
$$\textbf{Case } t = 2 : \text{Either } P(x^0) = P(x^1) = 0 \text{ or } P(x^0) = P(x^1) \wedge m^0 = m^1.$$

Through the paper, we say Π is CPA-1-sided (resp. CPA-2-sided) secure *without collusions* if $|\mathcal{Q}_{\mathsf{KGen}}| = 1$, i.e., the adversary cannot get more than one decryption key.[18]

Remark 2. PE schemes, satisfying CPA security (Definition 8) or CPA-1-sided security (Definition 9), can be built from different assumptions. Notably, [30] proposes an LWE-based PE scheme satisfying CPA-1-sided (and thus CPA) selective security, i.e., the adversary chooses the challenge messages and predicate inputs before receiving the master public key. By using complexity leveraging, the same construction achieves adaptive security (i.e., Definitions 8, 9) but this requires sub-exponential LWE.

4. Multi-key and Multi-input Predicate Encryption

We provide the formal definitions of multi-key PE and multi-input PE in the following Sects. 4.1 and 4.2, respectively. In Sect. 4.3, we show the relations between multi-key PE and multi-input PE schemes.

[18] For the sake of clarity, we implicitly assume that challenge messages and inputs have the same length, i.e., $|m^0| = |m^1|$ and $|x^0| = |x^1|$ (this is required to exclude trivial attacks). We stress that when the single-input PE scheme has an apriori bound on the length of the messages and attributes (defined on setup), the latter have same length by definition.

$\mathbf{G}^{\mathsf{CPA}\text{-}t\text{-}\mathsf{kPE}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{mpk}, \mathsf{msk}_1, \ldots, \mathsf{msk}_n) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$

$(m^0, m^1, x^0, x^1, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{KGen}(\mathsf{msk}_1,\cdot),\ldots,\mathsf{KGen}(\mathsf{msk}_n,\cdot)}(1^\lambda, \mathsf{mpk})$

$b \leftarrow\!\!\$\ \{0,1\}, c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, x^b, m^b)$

$b' \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk}_1,\cdot),\ldots,\mathsf{KGen}(\mathsf{msk}_n,\cdot)}(1^\lambda, c, \alpha)$

If $(b' = b)$**: return** 1

Else: return 0

Fig. 4. Game defining CPA-t-sided security of n-key PE.

4.1. *Multi-key PE*

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

$\mathsf{Setup}(1^\lambda)$: Upon input the security parameter 1^λ the setup algorithm outputs the master public key 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 ith master secret key msk_i and the ith index $v_i \in \mathcal{V}_i$. The algorithm outputs the ith secret decryption key 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 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.

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 $P_{v_1,\ldots,v_n}(x) = 1$.

Definition 10. (*Correctness of n-key PE*). A n-key PE with message space $\mathcal{M}$, input space $\mathcal{X}$, predicate space $\mathcal{P} = \{P_{v_1,\ldots,v_n}\}_{v_1,\ldots,v_n\in\mathcal{V}}$ indexed by $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$, is correct if $\forall\lambda \in \mathbb{N}, \forall m \in \mathcal{M}, \forall x \in \mathcal{X}, \forall(v_1, \ldots, v_n) \in \mathcal{V}$ such that $P_{v_1,\ldots,v_n}(x) = 1$, we have:

$$\mathbb{P}\left[\mathsf{Dec}(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}, \mathsf{Enc}(\mathsf{mpk}, x, m)) = m\right] \geq 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{mpk}, \mathsf{msk}_1, \ldots, \mathsf{msk}_n) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, v_i)$ for $i \in [n]$. The above probability is taken over the random coins of Setup, KGen, and Enc.

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 $P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 0$. 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., $P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 1$ and $m^0 = m^1$. These security definitions are formalized below.

Definition 11. (*CPA-1-sided and CPA-2-sided security of n-key PE*). Let $t \in [2]$. We say that a n-key PE Π is CPA-t-sided secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}_{\Pi,\mathsf{A}}^{\text{CPA-}t\text{-kPE}}(\lambda)$ is depicted in Fig. 4. Adversary 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[19]

$$\begin{aligned} &\textbf{Case } t = 1 : P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 0. \\ &\textbf{Case } t = 2 : \text{Either } P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 0 \\ &\qquad\qquad \text{or } P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) \wedge m^0 = m^1.^{19} \end{aligned}$$

4.2. *Multi-input PE*

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

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

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

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$: Let $i \in [n]$. The randomized encryption algorithm takes as input an encryption key 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.

$\mathsf{Dec}(\mathsf{dk}_P, c_1, \ldots, c_n)$: The deterministic decryption algorithm takes as input a secret decryption key dk_P for predicate $P \in \mathcal{P}$ 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 $P(x_1, \ldots, x_n) = 1$.

[19] As usual, we implicitly assume that challenge messages and inputs have the same length, i.e., $|m^0| = |m^1|$ and $|x^0| = |x^1|$ (this is required to exclude trivial attacks). We stress that when the multi-key PE scheme has an apriori bound on the length of the messages and attributes (defined on setup), the latter have same length by definition.

Definition 12. (*Correctness of n-input PE*). An n-input PE with message space $\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_n$, input space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, predicate space $\mathcal{P}$, is correct if $\forall \lambda \in \mathbb{N}$, $\forall (m_1, \ldots, m_n) \in \mathcal{M}$, $\forall (x_1, \ldots, x_n) \in \mathcal{X}$,$\forall P \in \mathcal{P}$ such that $P(x_1, \ldots, x_n) = 1$, we have:

$$\mathbb{P}\left[\mathsf{Dec}(\mathsf{dk}_P, c_1, \ldots, c_n) = (m_1, \ldots, m_n)\right] \geq 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$, $\mathsf{dk}_P \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, P)$, and $c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$ for $i \in [n]$. The above probability is taken over the random coins of Setup, KGen, and Enc.

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 dk_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 ek_i. Intuitively, the knowledge of ek_i impacts the validity condition that the adversary must satisfy (e.g., the challenge ciphertext cannot be decrypted). Indeed, ek_i would allow the adversary to produce arbitrary ith ciphertexts on arbitrary ith 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 dk_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 $P(x_1, \ldots, x_n) = 0$ (otherwise part of the challenge ciphertext can be decrypted).
- *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 ith encryption key 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 $P(x_1, \ldots, x_n) = 0$.

$\mathbf{G}^{\ell\text{-CPA-}t\text{-iPE}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$

$((m_i^0)_{i\in[n]}, (m_i^1)_{i\in[n]}, (x_i^0)_{i\in[n]}, (x_i^1)_{i\in[n]}, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Corr}(\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[n]}}(1^\lambda)$

$b \leftarrow\!\!\$\ \{0,1\}, c_1 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_1, x_1^b, m_1^b), \ldots, c_n \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_n, x_n^b, m_n^b)$

$b' \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Corr}(\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[n]}}(1^\lambda, c_1, \ldots, c_n, \alpha)$

If $(b' = b)$**: return** 1

Else: return 0

Fig. 5. Game defining CPA-t-sided security of n-input PE in the ℓ-corruptions setting. Oracle $\mathsf{Corr}(j)$ returns ek_j for $j \in [n]$.

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

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

where game $\mathbf{G}^{\ell\text{-CPA-}t\text{-iPE}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 5. Let $\mathcal{Q}_i = \{x | \exists (x, m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_i,\cdot,\cdot)}\}$ for $i \in [n] \backslash \mathcal{Q}_{\mathsf{Corr}}$ and $\mathcal{Q}_i = \mathcal{X}_i$ for $i \in \mathcal{Q}_{\mathsf{Corr}}$. Moreover, let $\mathcal{Q}_i^d$ (for $d \in \leftarrow\!\!\$$) be the ordered set 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 $j \in [k_i]$.[20] Adversary A is called valid if $|\mathcal{Q}_{\mathsf{Corr}}| \leq \ell$ and $\forall P \in \mathcal{Q}_{\mathsf{KGen}}$, $\forall j \in [n]$, $\forall i_1 \in [k_1 + 1], \ldots, \forall i_n \in [k_n + 1]$, we have

$$\begin{aligned}
&\textbf{Case } t = 1 : P(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) \\
&\qquad = P(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) = 0.
\end{aligned}$$

Case $t = 2$: Either

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

or

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

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

$\mathbf{G}^{\ell\text{-hyb-CPA-1-iPE}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{ek}_1,\ldots,\mathsf{ek}_n,\mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda,1^\ell),\ \mathsf{pub}=(\mathsf{ek}_{n-\ell+1},\ldots,\mathsf{ek}_n)$

$((m_i^0)_{i\in[n]},(m_i^1)_{i\in[n]},(x_i^0)_{i\in[n]},(x_i^1)_{i\in[n]},\alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{KGen}(\mathsf{msk},\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[n-\ell]}}(1^\lambda,\mathsf{pub})$

$b \leftarrow\!\!\$\ \{0,1\}, c_1 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_1,x_1^b,m_1^b),\ldots,c_n \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_n,x_n^b,m_n^b)$

$b' \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[n-\ell]}}(1^\lambda,c_1,\ldots,c_n,\alpha)$

If $(b'=b)$**: return** 1

Else: return 0

Fig. 6. Game defining CPA-t-sided security of n-input PE in the ℓ-hybrid setting.

Through the paper, for $t \in [2]$, we say that Π is CPA-t-sided secure in the ℓ-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 Π is CPA-t-sided secure in the *secret-key setting*. In case of both restrictions, we say that Π is CPA-t-sided secure in the *secret-key setting* and *without collusions* (i.e., $|\mathcal{Q}_{\mathsf{Corr}}| = 0$ and $|\mathcal{Q}_{\mathsf{KGen}}| = 1$).[21]

4.3. *Relating Multi-key PE and Multi-input PE*

Here, we show a construction of n-key PE from $(n+1)$-input PE supporting arbitrary predicates and tolerating 1 corruption. In more details, it suffices that the $(n+1)$-input PE satisfies a weaker flavor of security under corruptions, named ℓ-*hybrid setting* (which is formalized in this section).

Multi-input PE in the ℓ-Hybrid Setting. A multi-input PE in the hybrid setting allows generating (during setup) some encryptions keys that can be made public. The main difference between the hybrid setting and the corruption setting is that in the former the setup needs to know a priori which ones will be public (in other words, the setup depends on the keys that the adversary wants to leak/obtain). For this reason, it is easy to see that the hybrid setting is stronger than the secret-key one but significantly weaker than the corruption setting (in which the keys are leaked by the adversary in an adaptively fashion).

We assume that the Setup algorithm takes as input an additional parameter 1^ℓ denoting the number of keys that will be made public. Without loss of generality, we assume that the first $n-\ell$ keys $(\mathsf{ek}_1,\ldots,\mathsf{ek}_{n-\ell})$ are kept secret whereas the last ℓ keys $(\mathsf{ek}_{n-\ell+1},\ldots,\mathsf{ek}_n)$ are published. Observe that, for $\ell = 0$, the hybrid setting corresponds to the secret-key setting (see Sect. 4.2).

[21] As for multi-key PE, we implicitly assume that challenge messages and inputs have the same length, i.e., $|m_i^0| = |m_i^1|$ and $|x_i^0| = |x_i^1|$ for $i \in [n]$ (this is required to exclude trivial attacks). We stress that when the multi-input PE scheme has an apriori bound on the length of the messages and attributes (defined on setup), the latter have same length by definition.

Definition 14. (*ℓ-Hybrid CPA-1-sided and CPA-2-side security of n-input PE*). Let $t \in [2]$. We say that a n-input PE Π is CPA-t-sided secure in the ℓ-hybrid setting if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}^{\ell\text{-hyb-CPA-1-iPE}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 6. Let $\mathcal{Q}_i = \{x | \exists (x, m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_i,\cdot,\cdot)}\}$ for $i \in [n-\ell]$ and $\mathcal{Q}_i = \mathcal{X}_i$ for $i \in [n] \backslash [n-\ell]$. Moreover, let $\mathcal{Q}^d_i$ (for $d \in \{\leftarrow, \$\}$) be the ordered set composed of the predicate inputs $\mathcal{Q}_i$ and the challenge input x^d_i, i.e., $\mathcal{Q}^d_i = \{x^{(1,d)}_i, \ldots, x^{(k_i,d)}_i, x^{(k_i+1,d)}_i = x^d_i\}$ where $k_i = |\mathcal{Q}_i|$ and $x^{(j,d)} \in \mathcal{Q}_i$ for $j \in [k_i]$. Adversary A is called valid if $|\mathcal{Q}_{\mathsf{Corr}}| \leq \ell$ and $\forall P \in \mathcal{Q}_{\mathsf{KGen}}$, $\forall j \in [n]$, $\forall i_1 \in [k_1 + 1], \ldots, \forall i_n \in [k_n + 1]$, we have[22]

$$\begin{aligned}
&\textbf{Case } t = 1 : P(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) \\
&\qquad = P(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) = 0. \\
&\textbf{Case } t = 2 : \text{Either} \\
&\qquad P(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) \\
&\qquad = P(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) = 0 \\
&\qquad \text{or} \\
&\qquad P(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) \\
&\qquad = P(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) \wedge m_j^0 = m_j^1.
\end{aligned}$$

Multi-key PE from Multi-input PE. Here, we build a n-key PE from $(n+1)$-input PE that tolerates 1 public encryption key, i.e., 1-hybrid setting (Definition 14). The idea is to use the first n inputs of the predicate $P(x_1, \ldots, x_{n+1})$ (of $(n+1)$-input PE) to determine the indexes $(v_1, \ldots, v_n) \in \mathcal{V}$ that define the predicate $P_{v_1,\ldots,v_n}(x)$ of the n-key PE, i.e., $P(x_1, \ldots, x_{n+1}) = P(v_1, \ldots, v_n, x) = P_{v_1,\ldots,v_n}(x)$ where $x_i = v_i$ for $i \in [n]$ and $x_{n+1} = x$.

Construction 1. *Let* $\mathsf{iPE} = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *be a* $(n+1)$*-input PE scheme with message space* $\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_{n+1}$*, input space* $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_{n+1}$*, and predicate space* $\mathcal{P}_1 = \{P(x_1, \ldots, x_{n+1})\}$ *such that*

$$P(x_1, \ldots, x_{n+1}) = P_{x_1,\ldots,x_n}(x_{n+1}),$$

[22] As usual, we implicitly assume that the challenge messages and inputs have the same length, i.e., $|m^0_i| = |m^1_i|$ and $|x^0_i| = |x^1_i|$ for $i \in [n]$.

where $x_i \in \mathcal{X}_i$ *for* $i \in [n+1]$. *We build a* n*-key PE scheme with message space* $\mathcal{M} = \mathcal{M}_{n+1}$, *input space* $\mathcal{X} = \mathcal{X}_{n+1}$, *and predicate space* $\mathcal{P} = \{P_{v_1,\ldots,v_n}(x)\}_{(v_1,\ldots,v_n)\in\mathcal{V}}$ *indexed by* $\mathcal{V} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, *in the following way:*

- $\mathsf{Setup}(1^\lambda)$*: Upon input the security parameter* 1^λ *the randomized setup algorithm outputs* $\mathsf{mpk} = \mathsf{ek}_n$ *and* $\mathsf{msk}_1 = (\mathsf{ek}_1, \mathsf{dk}_P), \ldots, \mathsf{msk}_n = (\mathsf{ek}_n, \mathsf{dk}_P)$ *where* $(\mathsf{mpk}', \mathsf{ek}_1, \ldots, \mathsf{ek}_{n+1}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$ *and* $\mathsf{dk}_P \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}', P)$ *for* $P \in \mathcal{P}_1$.
- $\mathsf{KGen}(\mathsf{msk}_i, v_i)$*: Let* $i \in [n]$. *Upon input the* i*th master secret key* $\mathsf{msk}_i = (\mathsf{ek}_i, \mathsf{dk}_P)$, *and the* i*th predicate index* $v_i \in \mathcal{X}_i$, *the randomized key generator outputs* $\mathsf{dk}_{v_i} = (c_{v_i}, \mathsf{dk}_P)$ *where* $c_{v_i} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{ek}_i, v_i, \perp)$.
- $\mathsf{Enc}(\mathsf{mpk}, x, m)$*: Upon input the master public key* $\mathsf{mpk} = \mathsf{ek}_{n+1}$, *an input* $x \in \mathcal{X}_{n+1}$, *and a message* $m \in \mathcal{M}_{n+1}$, *the randomized encryption algorithm computes* $c \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{ek}_{n+1}, x, m)$.
- $\mathsf{Dec}(\mathsf{dk}_{v_1}, \ldots, \mathsf{dk}_{v_n}, c)$*: Upon input* n *secret decryption keys* $\mathsf{dk}_{v_1} = (c_{v_1}, \mathsf{dk}_P), \ldots, \mathsf{dk}_{v_n} = (c_{v_n}, \mathsf{dk}_P)$ *and a ciphertext* c, *the deterministic decryption algorithm outputs* m_{n+1} *where* $(m_1, \ldots, m_{n+1}) = \mathsf{Dec}_1(\mathsf{dk}_P, c_{v_1}, \ldots, c_{v_n}, c)$.

Correctness follows from the correctness of iPE. As for security, we establish the following result.

Theorem 4. *Let* iPE *be as above. For* $t \in [2]$, *if* iPE *is CPA-*t*-sided secure in the* 1*-hybrid model without collusions* (Definition 14) *then the* n*-key PE scheme* Π *from Construction* 1 *is CPA-*t*-sided secure (Definition* 11*).*

Proof. (CPA-1-sided security of Π) *Without loss of generality, we assume that the adversary* A *submits (at least) one query to each key generation oracle* $\mathsf{KGen}(\mathsf{msk}_1, \cdot), \ldots, \mathsf{KGen}(\mathsf{msk}_n, \cdot)$ *(proving the security of* Π *against this adversary implies the security of* Π *against any other adversary that does not query an oracle* $\mathsf{KGen}(\mathsf{msk}_j, \cdot, \cdot)$ *for a* $j \in [n]$*). Suppose there exists a valid PPT adversary* A *with a non-negligible advantage in breaking the CPA-*1*-sided security of* Π. *We build an adversary* A′ *that breaks the* 1*-hybrid CPA-*1*-side security (without collusions) of* iPE. A′ *is defined as follows:*

1. *Receive* ek_{n+1} *from the challenger and send it to* A.
2. *Send the query* P *(i.e., the predicate supported by* iPE*) to the* KGen_1 *oracle and receive* dk_P.
3. *item:answerspsqueriesspsmultispskeyspsPEspsfromspsmultispsinputspsPE Initialize* $\mathcal{L}_i = \{\emptyset\}$ *for* $i \in [n]$. A′ *answers to the incoming oracle queries as follows:*
   - *On input* $v_i \in \mathcal{X}_i$ *for* $\mathsf{KGen}(\mathsf{msk}_i, \cdot)$, *forward the query* $(v_i, \perp)$ *to oracle* $\mathsf{Enc}_1(\mathsf{ek}_i, \cdot, \cdot)$ *and receive the answer* c_{v_i}. *Add* v_i *to* $\mathcal{L}_i$ *and return* $\mathsf{dk}_{v_i} = (c_{v_i}, \mathsf{dk}_P)$.
4. *Receive the challenge* (m^0, m^1, x^0, x^1) *from* A. A′ *sends the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *where* $m_1^i = \ldots = m_n^i = \perp$, $m_{n+1}^i = m^i$, $x_j^i = x_j^{1-i} = x_{\bar{j}} \leftarrow\!\!\$\ \mathcal{L}_j$, *and* $x_{n+1}^i = x^i$, *for* $j \in [n]$ *and* $i \in \leftarrow\!\!\$$.
5. *Receive the challenge ciphertexts* $c_1, \ldots, c_{n+1}$ *and forward* c_{n+1} *to* A.

6. *Answer to the incoming oracle queries as in Item* 3.
7. *Return the output of* A.

Let d *be the challenge bit sampled by the challenger.* A' *perfectly simulates the view of* A. *Moreover, since* A *is a valid adversary, we have that* $\forall v_1 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)}, \ldots, \forall v_n \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_n,\cdot)}$, *we have* $P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 0$. *In order to be valid,* A' *needs to satisfy the condition of Definition* 14. *Let* $\mathcal{Q}_i^b$ *as defined in Definition* 14. *First, note that, for* $i \in [n]$, *we have that* $\mathcal{Q}_i^0 = \mathcal{Q}_i^1 = \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_i,\cdot)} = \mathcal{L}_i$ *since* $x_i^d = x_i^{1-d} = \bar{x}_i$ *are sampled from* $\mathcal{L}_i$ *(i.e.,* $\mathcal{Q}_i$ *does not contain any value that depends on the challenge bit* d*). Hence, the only case in which the adversary* A' *may evaluate the predicate* P *on an input that depends on the challenge bit* d *(i.e., the cases captured by Definition* 14*) is when* A' *uses the challenge ciphertext* c_{n+1}. *However, when* c_{n+1} *is used, the validity of* A *implies that* $\forall (v_1, \ldots, v_n) \in \mathcal{Q}_1^b \times \cdots \times \mathcal{Q}_n^b$ *(recall* $\mathcal{Q}_i^b = \mathcal{Q}_i^{1-b}$ *for* $i \in [n]$*),*

$$P(v_1, \ldots, v_n, x_{n+1}^0) = P_{v_1,\ldots,v_n}(x^0) = P_{v_0,\ldots,v_n}(x^1) = P(v_1, \ldots, v_n, x_{n+1}^1) = 0,$$

where $x_{n+1}^i = x^i$ *for* $i \in \leftarrow\!\!\$$. *Hence,* A' *submits only a single query to oracle* KGen_1 *and is also a valid adversary for* $\mathbf{G}_{\mathsf{iPE},\mathsf{A}'}^{\ell\text{-hyb-CPA-1-iPE}}(\lambda)$. *This concludes the proof.*

(CPA-2-sided security of Π) *The reduction is identical. The only difference is the analysis of the validity of* A'. *By definition* A *is a valid adversary with respect to the CPA-2-sided security of* iPE, *i.e.,* $\forall v_1 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)}, \ldots, \forall v_n \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_n,\cdot)}$, *we have*

$$\begin{aligned} &\textit{Either } P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = 0 \textit{ or} \\ &P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) \wedge m^0 = m^1. \end{aligned}$$

If A *satisfies the first part of the above condition, then the analysis of* A*'s validity is identical to that of CPA-1-sided security. On the other hand, if* A *satisfies the second part of the above condition, then the validity of* A *follows by using an similar argument to that of CPA-1-sided security and, in addition, observing that*

$$P(v_1, \ldots, v_n, x_{n+1}^0) = P_{v_1,\ldots,v_n}(x^0) = P_{v_1,\ldots,v_n}(x^1) = P(v_1, \ldots, v_n, x_{n+1}^1),$$

and $m_{n+1}^0 = m^0 = m^1 = m_{n+1}^1$. *This concludes the proof.* □

5. Constructions

In this section, we give different constructions of multi-key and multi-input PE (see also Sect. 1.2) for predicates $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \ldots \wedge P_n(x_n)$.

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 Sects. 5.2 and 5.3, we give two constructions of n-input PE from single-input PE, lockable obfuscation, and SKE/PKE. The first handles $\mathsf{poly}(\lambda)$-arity and it is CPA-1-side

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

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

$\mathbb{C}_{c,\mathsf{k}}(c_1, \ldots, c_{n-1}, \mathsf{dk}_P)$

Initialize: $\mathsf{k}_1 = \mathsf{k}, c_n = c$
For i **from** n **to** 1 **do:**
  $\mathsf{Dec}_1(\mathsf{dk}_P, \mathsf{Dec}_2(\mathsf{k}_i, c_i)) = v_i$
  If $v_i = \perp$**: return** $\perp$
  Else: let $v_i = (y_i, \mathsf{k}_{i+1})$
end for.
return y_n **where** $v_n = (y_n, \mathsf{k}_1)$

Fig. 7. On the left, the definition of the circuit $\mathbb{C}_c$ of Construction 2. On the right, the definition of the circuit $\mathbb{C}_{c,\mathsf{k}}$ of Construction 3.

secure without collusions and in the secret-key setting. The second handles $O(1)$-arity and it is CPA-1-side secure without collusions and in the $(n-1)$-corruptions setting. This second construction leverages a new nesting execution technique of (lockable obfuscated) circuits.

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 ith predicate P_i we have $P_i(x_i^\star) = 1$ (in Sect. 5.4 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 (in case of no corruptions, our CPA-2-sided secure multi-input Construction 3 supports $n = O(\log(\lambda))$).

5.1. *Multi-key PE from PE and Lockable Obfuscation*

Construction 2. *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* $\mathcal{P}_i = \{P_v(x)\}_{v \in \mathcal{V}_i}$ *indexed by* $\mathcal{V}_i$*. Without loss of generality, we assume that* PE_i *has ciphertext space* $\mathcal{Y}_i$*,* $\mathcal{M}_1 = \leftarrow\!\$^{m(\lambda)}$*, and* $\mathcal{M}_i = \mathcal{Y}_{i-1}$ *for every* $i \in [n]\backslash\{1\}$*. 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* $s \leftarrow\!\$ \leftarrow\!\$^{\lambda}$*).*[23]
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. 7, 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* Π *with message space* $\mathcal{M}$*, input space* $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$*, and predicate space* $\mathcal{P} = \{P_{v_1,\ldots,v_n}(x_1, \ldots, x_n) = P_{v_1}(x_1) \wedge \cdots \wedge P_{v_n}(x_n)\}_{(v_1,\ldots,v_n) \in \mathcal{V}}$ *indexed by* $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$ *(and* $P_{v_i} \in \mathcal{P}_i$ *for* $i \in [n]$*), as follows:*

[23] We stress each PE_i requires a different PRG with a particular stretch depending on the size of the message we wish to encrypt. Alternatively, it is possible to use a single PRG if the latter has an arbitrary polynomial stretch.

$\mathsf{Setup}(1^\lambda)$: *Upon input the security parameter* 1^λ *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* msk_i *and the* i*th predicate index* $v_i \in \mathcal{V}_i$, *the randomized key generator outputs* $\mathsf{dk}_{v_i} \leftarrow\!\$\ \mathsf{KGen}_i(\mathsf{msk}_1, P_{v_i})$ *where* $P_{v_i} \in \mathcal{P}_i$.

$\mathsf{Enc}(\mathsf{mpk}, x, m)$: *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* $y \leftarrow\!\$\ \leftarrow\!\$^{s(\lambda)}$ *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.

Theorem 5. *Let* $n = \mathit{poly}(\lambda)$, $\mathsf{PE}_1, \ldots, \mathsf{PE}_n$ *and* LOBF *be as above.*

1. *If each* $\mathsf{PE}_1, \ldots, \mathsf{PE}_n$ *is CPA secure (Definition 8) and* LOBF *is secure (Definition 2), then the* n*-key PE scheme* Π *from Construction 2 is CPA-*1*-sided secure (Definition 11).*
2. *If each* $\mathsf{PE}_1, \ldots, \mathsf{PE}_n$ *is CPA-*2*-sided secure (Definition 9) and* LOBF *is secure (Definition 2), then the* n*-key PE scheme* Π *from Construction 2 is CPA-*2*-sided secure (Definition 11).*

5.1.1. *Proof of Theorem 5*

CPA-1-sided security of Π (Theorem 5). Consider the predicate space $\mathcal{P}$ of Construction 2, i.e.,

$$\begin{aligned}\mathcal{P} &= \{P_{v_1,\ldots,v_n}(x_1, \ldots, x_n)\}_{(v_1,\ldots,v_n)\in\mathcal{V}} \\ &= \{P_{v_1}(x_1) \wedge \ldots \wedge P_{v_n}(x_n)\}_{(P_{v_1},\ldots,P_{v_n})\in\mathcal{P}_1\times\ldots\times\mathcal{P}_n}. \end{aligned} \tag{3}$$

Also, consider the validity condition of $\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}kPE}}_{\Pi,\mathsf{A}}$ (Definition 11). We can write such a validity condition for the predicate space $\mathcal{P}$ as follows: $\forall v_1 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)}, \ldots, v_n \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_n,\cdot)}$,

$$\begin{aligned}&P_{v_1,\ldots,v_n}(x_1^0, \ldots, x_n^0) = P_{v_1,\ldots,v_n}(x_1^1, \ldots, x_n^1) \\ &= \left(P_{v_1}(x_1^0) \wedge \ldots \wedge P_{v_n}(x_n^0)\right) = 0 \ \wedge\ \left(P_{v_1}(x_1^0) \wedge \ldots \wedge P_{v_n}(x_n^0)\right) = 0,\end{aligned}$$

where $x_0 = (x_1^0, \ldots, x_n^0)$ and $x_1 = (x_1^1, \ldots, x_n^1)$ are the two input challenges output by the adversary. The above equation can be rewritten as follows: $\exists j_0, j_1 \in [n], \forall v_{j_0} \in$

$\mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{j_0},\cdot)}$, $\forall v_{j_1} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{j_1},\cdot)}$,

$$P_{v_{j_0}}(x^0_{j_0}) = 0 \wedge P_{v_{j_1}}(x^1_{j_1}) = 0. \tag{4}$$

Hence, in order to be valid with respect to $\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}kPE}}_{\Pi,\mathsf{A}}$, A needs to satisfy the above equation. Let $\mathbf{Validity}_{j_0,j_1}$ the validity condition (as defined in Eq. (4)) with respect to some $j_0, j_1 \in [n]$. By taking into account the above point, the CPA-1-sided security of Construction 2 follows by proving the following lemma.

Lemma 1. *Let $j_0, j_1 \in [n]$. If both PE_{j_0} and PE_{j_1} are CPA secure (Definition 8) and LOBF is secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}kPE}}_{\Pi,\mathsf{A}}(\lambda) = 1 \middle| \mathbf{Validity}_{j_0,j_1}\right] - \frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *Consider the following hybrid experiments:*

- $\mathbf{H}^b_0(\lambda)$*: This is exactly the experiment* $\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}kPE}}_{\Pi,\mathsf{A}}$ *conditioned to* $\mathbf{Validity}_{j_0,j_1}$ *where the challenge bit is b, i.e., the adversary is valid and satisfies condition* $\mathbf{Validity}_{j_0,j_1}$.
- $\mathbf{H}^b_1(\lambda)$*: Same as* $\mathbf{H}^b_0$*, except that the challenger computes* $c_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_{j_b}(\mathsf{mpk}_{j_b}, x^b_{j_b}, w)$ *where* $w \leftarrow\!\!\$\ \mathcal{M}_{j_b}$ *(instead of* $c_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_{j_b}(\mathsf{mpk}_{j_b}, x^b_{j_b}, c_{j_b-1})$.
- $\mathbf{H}^b_2(\lambda)$*: Same as* $\mathbf{H}^b_1$*, except that the challenger simulates the challenge ciphertext* $c = \widetilde{\mathbb{C}}$ *using the simulator of the lockable obfuscation scheme* LOBF*, i.e.,* $\widetilde{\mathbb{C}} \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{C}_c|}, 1^{|m_b|})$.

Claim 1. $\mathbf{H}^b_0(\lambda) \approx_c \mathbf{H}^b_1(\lambda)$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}^b_0(\lambda)$ *and* $\mathbf{H}^b_1(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA security of* PE_{j_b}. A *is defined as follows:*

1. *Receive* mpk_{j_b} *from the challenger.*
2. *Send* $\mathsf{mpk} = (\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n)$ *to* D *where* $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow\!\!\$\ \mathsf{Setup}_i(1^\lambda)$ *for* $i \in [n]\backslash\{j_b\}$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $v_i \in \mathcal{V}_i$ *for* $\mathsf{KGen}(\mathsf{msk}_i, \cdot)$, A *proceeds as follows: If* $j_b = i$*, it forwards the query* $P_{v_i} \in \mathcal{P}_{j_b}$ *to* KGen_{j_b} *and returns the answer* dk_{v_i}*. Otherwise (if* $j_b \neq i$*), it returns* $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_i, P_{v_i})$ *for* $P_{v_i} \in \mathcal{P}_i$.
4. *Receive the challenge* $(m^0, m^1, (x^0_1, \ldots, x^0_n), (x^1_1, \ldots, x^1_n))$ *from* D.
5. *Sample* $y \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and set* $c_0 = y$.
6. *For* $i \in [j_b - 1]$*, compute* $c_i \leftarrow\!\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x^b_i, c_{i-1})$.
7. *Send the challenge* $(m^0_*, m^1_*, x^b_{j_b})$ *where* $m^0_* = c_{j_b}$, $m^1_* \leftarrow\!\!\$\ \mathcal{M}_{j_b}$ *and receive the challenge ciphertext* c^*.
8. *For* $i \in [n] \setminus [j_b]$*, compute* $c_i \leftarrow\!\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x^b_i, c_{i-1})$ *where* $c_{j_b} = c^*$.
9. *Finally, send* $c = \widetilde{\mathbb{C}} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_n}, y, m^b)$ *to* D.

10. *Answer to the incoming oracle queries as in Item* 3.
11. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if $d = 0$,* A *simulates* $\mathbf{H}_0^b(\lambda)$. *On the other hand, if $d = 1$,* A *simulates* $\mathbf{H}_1^b(\lambda)$. *In addition, since* D *is valid and satisfies the condition* $\mathbf{Validity}_{j_0,j_1}$, *we conclude that* $\forall v_{j_0} \in \mathcal{V}_{j_0}$, $P_{v_{j_0}}(x_{j_b}^0) = 0$. *This implies that* $\forall P \in \mathcal{Q}_{\mathsf{KGen}_{j_b}}$, $P(x_{j_b}^0) = 0$. *Hence,* A *is a valid adversary with the same advantage of* D. *This concludes the proof.* □

Claim 2. $\mathbf{H}_1^b(\lambda) \approx_c \mathbf{H}_2^b(\lambda)$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_1^b(\lambda)$ *and* $\mathbf{H}_2^b(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of lockable obfuscation* LOBF. A *is defined as follows:*

1. *Send* $\mathsf{mpk} = (\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n)$ *to* D *where* $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow\!\!\$\ \mathsf{Setup}_i(1^\lambda)$ *for* $i \in [n]$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $v_i \in \mathcal{V}_i$ *for* $\mathsf{RKGen}(\mathsf{msk}_i, \cdot)$, A *returns* $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_i, P_{v_i})$ *for* $P_{v_i} \in \mathcal{P}_i$.
3. *Receive the challenge* $(m^0, m^1, (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For* $i \in [n] \setminus [j_b]$, *compute* $c_i \leftarrow\!\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x_i^b, c_{i-1})$ *where* $c_{j_b} \leftarrow\!\!\$\ \mathcal{M}_{j_b}$.
5. *The adversary* A *sends* $(\mathbb{C}_{c_n}, m^b)$ *to the challenger and receives back the obfuscated circuit* $\widetilde{\mathbb{C}}$ *from the challenger.*
6. A *returns* $c = \widetilde{\mathbb{C}}$ *to* D.
7. *Answer to the incoming oracle queries as in Item* 2.
8. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. When $d = 0$, A *simulates* $\mathbf{H}_1^b(\lambda)$; *otherwise, if $d = 1$,* A *simulates* $\mathbf{H}_2^b(\lambda)$. *Thus,* A *has the same non-negligible advantage of* D *with respect to the experiment* $\mathbf{G}^{\mathsf{lock\text{-}sim}}_{\mathsf{LOBF},\mathsf{A},\mathsf{S}}(\lambda)$. *This concludes the proof.* □

Claim 3. $\mathbf{H}_2^b(\lambda) \equiv \mathbf{H}_2^{1-b}(\lambda)$.

Proof. *The claim follows by observing that these experiments do not depend on the challenge bit b.* □

Lemma 1 follows by combining Claims 1–3. □

By leveraging Lemma 1, we conclude that Π of Construction 2 is CPA-1-sided secure.

CPA-2-sided security of Π (Theorem 5). Consider the validity condition of $\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}kPE}}_{\Pi,\mathsf{A}}$ (Definition 11). This can be rewritten with respect to the definition of $\mathcal{P}$ (Eq. (3)) as follows: $\exists j_0, j_1 \in [n]$, $\forall v_{j_0} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{j_0},\cdot)}$, $\forall v_{j_1} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{j_1},\cdot)}$, $\forall (v_1, \ldots, v_n) \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)} \times \cdots \times \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_n,\cdot)}$,

$$\begin{aligned} &\text{Either } P_{v_{j_0}}(x_{j_0}^0) = 0 \wedge P_{v_{j_1}}(x_{j_1}^1) = 0 \\ &\quad \text{or } P_{v_1}(x_1^0) = P_{v_1}(x_1^1) \wedge \ldots \wedge P_{v_n}(x_n^0) = P_{v_n}(x_n^1) \wedge m_0 = m_1 \end{aligned} \tag{5}$$

Consider the following conditions:

$$\begin{aligned}\textbf{Validity}_{0,j_0,j_1} : &\forall v_{j_0} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{j_0},\cdot)}, \forall v_{j_1} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{j_1},\cdot)},\\ &P_{v_{j_0}}(x^0_{j_0}) = 0 \wedge P_{v_{j_1}}(x^1_{j_1}) = 0\\ \textbf{Validity}_1 : &\forall (v_1,\ldots,v_n) \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)} \times \ldots \times \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_n,\cdot)},\\ &P_{v_1}(x^0_1) = P_{v_1}(x^1_1) \wedge \ldots \wedge P_{v_n}(x^0_n) = P_{v_n}(x^1_n) \wedge m^0 = m^1.\end{aligned}$$

By leveraging the above validity conditions we can rephrase Eq. (5) as follows: $\exists j_0, j_1 \in [n]$ such that

$$\text{Either } \textbf{Validity}_{0,j_0,j_1} \text{ or } \textbf{Validity}_1.$$

Hence, in order to be valid with respect to $\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}kPE}}_{\Pi,\mathsf{A}}$, A needs to satisfy the above equation. By taking into account the above point, the CPA-2-sided security of Construction 2 follows by proving the following lemmas.

Lemma 2. *Let $j_0, j_1 \in [n]$. If both PE_{j_0} and PE_{j_1} are CPA-2-sided secure (Definition 9) and LOBF is secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}kPE}}_{\Pi,\mathsf{A}}(\lambda) = 1 \middle| \textbf{Validity}_{0,j_0,j_1}\right] - \frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *The lemma follows by using an identical argument to that of Lemma 1.* □

Lemma 3. *If each $\mathsf{PE}_1,\ldots,\mathsf{PE}_n$ are CPA-2-sided secure (Definition 9), then*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}kPE}}_{\Pi,\mathsf{A}}(\lambda) = 1 \middle| \textbf{Validity}_1\right] - \frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *Consider the following hybrid experiments:*

$\mathbf{H}^b_0(\lambda)$*: This is exactly the experiment* $\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}kPE}}_{\Pi,\mathsf{A}}$ *conditioned to* $\textbf{Validity}_1$ *where the challenge bit is b, i.e., the adversary is valid and satisfies the condition* $\textbf{Validity}_1$.

$\mathbf{H}^b_i(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}^b_{i-1}$*, except that the challenger computes* $c_i \leftarrow\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x^{1-b}_i, c_{i-1})$ *(instead of* $c_i \leftarrow\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x^b_i, c_{i-1})$*).*

Claim 4. *For $i \in [n]$, $\mathbf{H}^b_{i-1}(\lambda) \approx_c \mathbf{H}^b_i(\lambda)$.*

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}^b_{i-1}(\lambda)$ *and* $\mathbf{H}^b_i(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA-2-sided security of* PE_i*.* A *is defined as follows:*

1. *Receive* mpk_i *from the challenger.*
2. *Send* $\mathsf{mpk} = (\mathsf{mpk}_1,\ldots,\mathsf{mpk}_n)$ *to* D *where* $(\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow\!\$\ \mathsf{Setup}_i(1^\lambda)$ *for* $j \in [n]\backslash\{i\}$.

3. A *answers to the incoming oracle queries as follows:*
   - *On input* $v_j \in \mathcal{V}_j$ *for* $\mathsf{RKGen}(\mathsf{msk}_j, \cdot)$, A *proceeds as follows: If* $j = i$, *it forwards the query* $P_{v_j} \in \mathcal{P}_i$ *to* KGen_i *and returns the answer* dk_{v_j}. *Otherwise (if* $j \neq i$*), it returns* $\mathsf{dk}_{v_j} \leftarrow\!\!\$\ \mathsf{KGen}_j(\mathsf{msk}_j, P_{v_j})$ *for* $P_{v_j} \in \mathcal{P}_j$.
4. *Receive the challenge* $(m^0, m^1, (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
5. *Sample* $y \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and set* $c_0 = y$.
6. *For* $j \in [i-1]$, *compute* $c_j \leftarrow\!\!\$\ \mathsf{Enc}_j(\mathsf{mpk}_j, x_j^{1-b}, c_{j-1})$.
7. *Send the challenge* $(m_*^0, m_*^1, x^0 = x_i^b, x^1 = x_i^{1-b})$ *where* $m_*^0 = m_*^1 = c_{i-1}$, *and receive the challenge ciphertext* c^*.
8. *For* $j \in [n] \setminus [i]$, *compute* $c_j \leftarrow\!\!\$\ \mathsf{Enc}_j(\mathsf{mpk}_j, x_j^b, c_{j-1})$ *where* $c_i = c^*$.
9. *Finally, send* $c = \widetilde{\mathbb{C}} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_n}, y, m_b)$ *to* D.
10. *Answer to the incoming oracle queries as in Item* 3.
11. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_{i-1}^b(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_i^b(\lambda)$. *In addition, since* D *is valid and satisfies the condition* $\mathbf{Validity}_1$, *we conclude that* $\forall v_i \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_i, \cdot)}$, $P_{v_i}(x_i^0) = P_{v_i}(x_i^1)$. *Hence,* A *is a valid adversary with the same advantage of* D. *This concludes the proof.* □

Claim 5. $\mathbf{H}_n^b(\lambda) \equiv \mathbf{H}_n^{1-b}(\lambda)$.

Proof. *Since* $\mathbf{Validity}_1$ *holds, we know that* $m_0 = m_1$. *Hence, these experiments are identically distributed.* □

Lemma 3 follows by combining Claims 4 and 5. □

By combining Lemmas 2 and 3 we conclude that Π of Construction 2 is CPA-2-sided secure.

5.2. *Secret-key Setting: Multi-input PE from PE, Lockable Obfuscation and SKE*

Secret-key setting. We present our n-input PE construction that is CPA-1-sided secure in the secret-key setting without collusions, for $n = \mathsf{poly}(\lambda)$. It leverages a CPA-1-sided secure single-input PE, lockable obfuscation, and SKE. The same construction is CPA-2-sided secure in the secret-key setting without collusions for $n = O(\log(\lambda))$, if the initial single-input PE is CPA-2-sided secure.

Construction 3. (n-input PE in the secret-key setting). *Consider the following primitives:*

1. *A PE scheme* $\mathsf{PE}_1 = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *with message space* $\mathcal{M}_1 = \leftarrow\!\!\$^{m(\lambda)} \times \mathcal{M}_1'$, *input space* $\mathcal{X}_1 = \mathcal{X}_{1,1} \times \cdots \times \mathcal{X}_{1,n}$, *and predicate space* $\mathcal{P}_1 = \{P(x_1, \ldots, x_n)\} = \{P_1(x_1) \wedge \cdots \wedge P_n(x_n)\}$. *Without loss of generality, we assume that* PE *has ciphertext space* $\mathcal{M}_2$ *and there exists a (single) wildcard input* $(x_1^\star, \ldots, x_n^\star) \in \mathcal{X}_1$ *such that* $\forall (P_1(x_1) \wedge \cdots \wedge P_n(x_n)) \in \mathcal{P}_1, \forall i \in [n]$, $P_i(x_i^\star) = 1$.

2. *A SKE scheme* $\mathsf{SKE} = (\mathsf{KGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ *with message space* $\mathcal{M}_2$*. Without loss of generality, we assume that* SKE *has key space* $\mathcal{M}'_1$
3. *A lockable obfuscation scheme* $\mathsf{LOBF} = (\mathsf{Obf}, \mathsf{Eval})$ *with message space* $\mathcal{M}_3$ *for the family of circuits* $\mathcal{C}_{n,s,d}(\lambda) = \{\mathbb{C}_{c,\mathsf{k}}\}$ *as defined in Fig. 7, where* $n(\lambda)$, $s(\lambda)$, $d(\lambda)$ *depends on the schemes* PE *and* SKE *used, and the circuit depth of the circuits* $\mathcal{C}_{n,s,d}(\lambda)$.

We build a n-input PE scheme with message space $\mathcal{M} = \overbrace{\mathcal{M}_3 \times \cdots \times \mathcal{M}_3}^{n}$*, input space* $\mathcal{X} = \mathcal{X}_1$*, and predicate space* $\mathcal{P} = \mathcal{P}_1 = \{P(x_1, \ldots, x_n)\} = \{P_1(x_1) \wedge \cdots \wedge P_n(x_n)\}$ *with wildcard (i.e., there exists a (single) wildcard* $(x_1^\star, \ldots, x_n^\star) \in \mathcal{X}$ *such that* $\forall (P_1(x_1) \wedge \cdots \wedge P_n(x_n)) \in \mathcal{P}, \forall i \in [n],\ P_i(x_i^\star) = 1$*), as follows:*

$\mathsf{Setup}(1^\lambda)$*: Upon input the security parameter* 1^λ*, the randomized setup algorithm outputs* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ *and* msk *where* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$, $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$, $\mathsf{k}_{n+1} = \mathsf{k}_1$*, and* $\mathsf{k}_i \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ *for* $i \in [n]$.

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

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$*: Let* $i \in [n]$*. Upon input an encryption key* $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$*, an input* $x_i \in \mathcal{X}_{1,i}$*, and a message* $m_i \in \mathcal{M}_3$*, the randomized encryption algorithm samples* $y_i \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and compute* $c_i^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_i, \mathsf{k}_{i+1}))$ *where* $x_j = x_j^\star$ *for any* $j \in [n] \backslash \{i\}$*. Finally, it outputs* $c = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$ *where* $\widetilde{\mathbb{C}}_i \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}, y_i, m_i)$ *and* $c_i^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_i, c_i^{(1)})$.

$\mathsf{Dec}(\mathsf{dk}_P, c_1, \ldots, c_n)$*: Upon input a secret decryption key* dk_P *for predicate* $P \in \mathcal{P}$*, and n ciphertexts* $(c_1, \ldots, c_n)$ *such that* $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$ *for* $i \in [n]$*. The deterministic decryption returns* $(m_1, \ldots, m_n)$ *where* $m_i = \mathsf{Eval}(\widetilde{\mathbb{C}}_i, (c_{i+1}^{(2)}, \ldots, c_n^{(2)}, c_1^{(2)}, \ldots, c_{i-1}^{(2)}, \mathsf{dk}_P))$ *for* $i \in [n]$.

As usual, correctness follows from the correctness of the underlying primitives. Below, we establish the following result.

Theorem 6. *Let* PE, SKE, *and* LOBF *be as above.*

1. *For* $n = \mathsf{poly}(\lambda)$*, if* PE *is CPA-1-sided secure without collusions (Definition 9),* SKE *is CPA secure (Definition 4), and* LOBF *is secure (Definition 2), then the n-input PE scheme* Π *from Construction 3 is CPA-1-sided secure in the secret-key setting without collusions (Definition 13).*
2. *For* $n = O(\log(\lambda))$*, if* PE *is CPA-2-sided secure without collusions (Definition 9),* SKE *is CPA secure (Definition 4), and* LOBF *is secure (Definition 2), then the n-input PE scheme* Π *from Construction 3 is CPA-2-sided secure in the secret-key setting without collusions (Definition 13).*

5.2.1. *Proof of Theorem 6*

CPA-1-sided security of Π (Theorem 6). Consider the predicate space $\mathcal{P} = \{P(x_1, \ldots, x_n)\}$ of Construction 3 where $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \cdots \wedge P_n(x_n)$. Let $P^* \in \mathcal{P}$ be the only predicate for which the adversary will ask the decryption key dk_{P^*} during

the experiment $\mathbf{G}^{\text{0-CPA-1-iPE}}_{\Pi,\mathsf{A}}$ (recall that we prove the security of Construction 3 in the scenario without collusions, i.e., $|\mathcal{Q}_{\mathsf{KGen}}| = 1$). Also, consider the validity condition of $\mathbf{G}^{\text{0-CPA-1-iPE}}_{\Pi,\mathsf{A}}$. We can write such a validity condition with respect to $P^* \in \mathcal{Q}_{\mathsf{KGen}} = \{P^*\}$ as follows: $\forall j \in [n], \forall i_1 \in [k_1+1], \ldots, \forall i_n \in [k_n+1]$,

$$
\begin{aligned}
&P^*(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) \\
&= P^*(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) \\
&= P_1^*(x_1^{(i_1,0)}) \wedge \cdots \wedge P_{j-1}^*(x_{j-1}^{(i_{j-1},0)}) \wedge P_j^*(x_j^0) \wedge P_{j+1}^*(x_{j+1}^{(i_{j+1},0)}) \wedge \cdots \wedge P_{i_n}^*(x_n^{(i_n,0)}) \\
&= P_1^*(x_1^{(i_1,1)}) \wedge \cdots \wedge P_{j-1}^*(x_{j-1}^{(i_{j-1},1)}) \wedge P_j^*(x_j^1) \wedge P_{j+1}^*(x_{j+1}^{(i_{j+1},1)}) \wedge \cdots \wedge P_n^*(x_n^{(i_n,1)}) = 0,
\end{aligned}
$$

where $\mathcal{Q}_i^b = \{x_i^{(1,b)}, \ldots, x_i^{(k_i,b)}, x_i^{(k_i+1,b)} = x_i^b\}$ is the ordered list composed of the k_i predicate inputs $\mathcal{Q}_i$ submitted to oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$ and the challenge input x_i^b (as defined in Definition 13). The above equation can be rewritten as follows: $\exists j_0, j_1 \in [n]$, $\forall (x_1', \ldots, x_n') \in \mathcal{Q}_1 \times \cdots \times \mathcal{Q}_n$,

$$
\begin{aligned}
&\Big(\Big(P_1^*(x_1^0) = 0 \wedge \cdots \wedge P_n^*(x_n^0) = 0\Big) \vee \Big(P_{j_0}^*(x_{j_0}^0) = 0 \wedge P_{j_0}^*(x_{j_0}') = 0\Big)\Big) \wedge \\
&\Big(\Big(P_1^*(x_1^1) = 0 \wedge \cdots \wedge P_n^*(x_n^1) = 0\Big) \vee \Big(P_{j_1}^*(x_{j_1}^1) = 0 \wedge P_{j_1}^*(x_{j_1}') = 0\Big)\Big). \qquad (6)
\end{aligned}
$$

Note that in the above equation we made explicit the challenge inputs and the inputs submitted to the encryption oracles. For this reason, it is enough to quantify over all $(x_1', \ldots, x_n') \in \mathcal{Q}_1 \times \cdots \times \mathcal{Q}_n$ where $\mathcal{Q}_i = \{x_i^{(1)}, \ldots, x_i^{(k_i)}\}$ are the inputs submitted to oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$. Hence, in order to be valid, A needs to satisfy the condition defined by Eq. (6). These conditions are defined by the events below: for some $j_0, j_1 \in [n]$,

$$
\begin{aligned}
&\textbf{Validity}_1 : \\
&\quad P_1^*(x_1^0) = 0 \wedge \cdots \wedge P_n^*(x_n^0) = 0 \wedge P_1^*(x_1^1) = 0 \wedge \cdots \wedge P_n^*(x_n^1) = 0. \\
&\textbf{Validity}_{2,j_0,j_1} : \forall x_{j_0}' \in \mathcal{Q}_{j_0}, \forall x_{j_1}' \in \mathcal{Q}_{j_1}, \\
&\quad P_{j_0}^*(x_{j_0}^0) = 0 \wedge P_{j_0}^*(x_{j_0}') = 0 \wedge P_{j_1}^*(x_{j_1}^1) = 0 \wedge P_{j_1}^*(x_{j_1}') = 0. \\
&\textbf{Validity}_{3,j_0} : \forall x_{j_0}' \in \mathcal{Q}_{j_0}, \\
&\quad P_{j_0}^*(x_{j_0}^0) = 0 \wedge P_{j_0}^*(x_{j_0}') = 0 \wedge P_1^*(x_1^1) = 0 \wedge \cdots \wedge P_n^*(x_n^1) = 0. \\
&\textbf{Validity}_{4,j_1} : \forall x_{j_1}' \in \mathcal{Q}_{j_1}, \\
&\quad P_1^*(x_1^0) = 0 \wedge \cdots \wedge P_n^*(x_n^0) = 0 \wedge P_{j_1}^*(x_{j_1}^1) = 0 \wedge P_{j_1}^*(x_{j_1}') = 0.
\end{aligned}
$$

For the sake of clarity, in the rest of this proof, we use the notation $\mathbb{V}_i \stackrel{\text{def}}{=} \mathbb{C}_{c_i^{(2)},\mathsf{k}_{i+1}}$ where $c_i^{(2)}$ and k_{i+1} will be clear from the context. Also, $[a:b]_n^+ = \{a, a+1, \ldots, n, 1, 2, \ldots, b\}$. If $1 \le a \le b \le n$, we have $[a:b]_n^+ = \{a, a+1, \ldots, b\}$. Similarly, $[a:b]_n^- = \{a, a-1, \ldots, 1, n, n-1, \ldots, b\}$. If $1 \le b \le a \le n$, we have $[a:b]_n^- = \{a, a-1, \ldots, b\}$.

Lemma 4. *If* PE *is CPA-1-sided secure without collusions (Definition 9) and* LOBF *is secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}^{\text{0-CPA-1-iPE}}_{\Pi,\mathsf{A}}(\lambda)=1 \wedge |\mathcal{Q}_{\mathsf{KGen}}|=1 \middle| \mathbf{Validity}_1\right]-\frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *Consider the following hybrid experiments:*

- $\mathbf{H}_0^{b,0}(\lambda)$*: This is exactly the experiment* $\mathbf{G}^{\text{0-CPA-1-iPE}}_{\Pi,\mathsf{A}}(\lambda)$ *conditioned to the event* $\mathbf{Validity}_1$ *where the challenge bit is b, i.e., the adversary is valid and satisfies the condition* $\mathbf{Validity}_1$*.*
- $\mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}_0^{b,i-1}$*, except that the challenger changes how it computes the challenger ciphertext* c_i*. The value* $c_i^{(1)}$ *challenge ciphertext* $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$ *is computed as* $c_i^{(1)} \leftarrow\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1,\ldots,x_n), 0^{s(\lambda)+k(\lambda)})$ *(instead of* $c_i^{(1)} \leftarrow\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1,\ldots,x_n), (y_i, \mathsf{k}_{i+1}))$*) where* $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$ *(for some function k),* $x_i = x_i^0$*, and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{i\}$*. Observe that* $c_i^{(1)}$ *is computed by fixing* $x_i = x_i^0$ *(instead of* $x_i = x_i^b$*), i.e., the input* $(x_1,\ldots,x_n)$ *used to compute the* i*th challenge ciphertext is fixed and does not depend on the challenge bit b.*
- $\mathbf{H}_1^{b,0}(\lambda)$*: Identical to* $\mathbf{H}_0^{b,n}(\lambda)$*.*
- $\mathbf{H}_1^{b,i}(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}_1^{b,i-1}$*, except that the challenger changes how it computes the challenger ciphertext* c_i*. Formally, the value* $\widetilde{\mathbb{C}}_i$ *of challenge ciphertext* $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$ *is simulated by the challenger using the simulator of the lockable obfuscation scheme* LOBF*, i.e.,* $\widetilde{\mathbb{C}}_i \leftarrow\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_i|}, 1^{|m_i^b|})$*.*

Claim 6. $\mathbf{H}_0^{b,i-1}(\lambda) \approx_c \mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [n]$*.*

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_0^{b,1-i}(\lambda)$ *and* $\mathbf{H}_0^{b,i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA-1-sided security without collusions of* PE*.* A *is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $\mathsf{k}_j \leftarrow\!\$\ \mathsf{KGen}_2(1^\lambda)$ *for* $j \in [n]$*. Let* $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$ *for* $i \in [n]$ *where* $\mathsf{k}_{n+1} = \mathsf{k}_1$*.*
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen*, forward the query* P^* *to* KGen_1 *and return the answer* dk_P*.*
   - *On input* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$*, return* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)}) \leftarrow\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$*.*
4. *Receive the challenge* $((m_1^0,\ldots,m_n^0),(m_1^1,\ldots,m_n^1),(x_1^0,\ldots,x_n^0),(x_1^1,\ldots,x_n^1))$ *from* D*.*
5. *For any* $j \in [n]$*,* A *proceeds as follows:*

   Case $j < i$*: Sample* $y_j \leftarrow\!\$ \leftarrow\!\$^{s(\lambda)}$*. Execute* $c_j^{(1)} \leftarrow\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1,\ldots,x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$*, and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$*.*

Case $j = i$: Send the challenge $(m_*^0 = (y_i, \mathsf{k}_{i+1}), m_*^1 = 0^{s(\lambda)+k(\lambda)}, x_*^0 = (x_{*1}^0, \ldots, x_{*n}^0), x_*^1 = (x_{*1}^1, \ldots, x_{*n}^1))$ where $y_i \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$, $x_{*i}^0 = x_i^b$, $x_{*i}^1 = x_i^0$, and $x_{*j}^0 = x_{*j}^1 = x_j^\star$ for $j \in [n]\backslash\{i\}$. Receive the challenge ciphertext c^* from the challenger. Set $c_i^{(1)} = c^*$.

Case $j > i$: Sample $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$ and compute $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ where $x_j = x_j^b$, and $x_{j'} = x_{j'}^\star$ for $j' \in [n]\backslash\{j\}$.

6. *Compute $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ where $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{ek}_j, c_j^{(1)})$ and $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ for any $j \in [n]$.*
7. *Send the challenge ciphertexts $(c_1, \ldots, c_n)$ to* D.
8. *Answer to the incoming oracle queries as in Item* 3.
9. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if $d = 0$,* A *simulates $\mathbf{H}_0^{b,i-1}(\lambda)$. On the other hand, if $d = 1$,* A *simulates $\mathbf{H}_0^{b,i}(\lambda)$. Moreover, conditioned to the event* **Validity**$_1$ *(i.e.,* D *satisfies* **Validity**$_1$*), we know that* D *asks for a single decryption key dk_{P^*} for P^* and $P_i^*(x_i^0) = 0 \wedge P_i^*(x_i^1) = 0$. Because of this,* A *submits a single query P^* to oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$ and it is also a valid adversary for the experiment $\mathbf{G}_{\mathsf{PE},\mathsf{A}}^{\mathsf{CPA\text{-}1\text{-}PE}}(\lambda)$ with the same advantage of* D. *This concludes the proof.* □

Claim 7. $\mathbf{H}_1^{b,i-1}(\lambda) \approx_c \mathbf{H}_1^{b,i}(\lambda)$ *for $i \in [n]$.*

Proof. Suppose there exists a PPT distinguisher D *that distinguishes between $\mathbf{H}_1^{b,1-i}(\lambda)$ and $\mathbf{H}_1^{b,i}(\lambda)$ with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF. A *is defined as follows:*

1. *Compute $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, \mathsf{k}_{j-1})$ for $j \in [n]$. Let $\mathsf{k}_{n+1} = \mathsf{k}_1$.*
2. A *answers to the incoming oracle queries as follows:*
   - *On input $P^* \in \mathcal{P}$ for* KGen*, return $\mathsf{dk}_{P^*} \leftarrow\$ \mathsf{KGen}(\mathsf{msk}, P^*)$.*
   - *On input $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ for $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ where $j \in [n]$, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)}) \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.*
3. *Receive the challenge $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ from* D.
4. *For any $j \in [n]$, compute $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ and $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [n]\backslash\{j\}$.*
5. *For any $j \in [n] \setminus \{i\}$,* A *proceeds as follows:*

   Case $j < i$: Compute $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^b|})$.

   Case $j = i$: Send the challenge $(\mathbb{V}_i, m_i^b)$ to the challenger and receive $\widetilde{\mathbb{C}}$. Set $\widetilde{\mathbb{C}}_i = \widetilde{\mathbb{C}}$.

   Case $j > i$: Compute $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ where $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$.

6. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
7. *Answer to the incoming oracle queries as in Item* 2.
8. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_1^{b,i-1}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_1^{b,i}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 8. $\mathbf{H}_1^{b,n}(\lambda) \equiv \mathbf{H}_1^{1-b,n}(\lambda)$.

Proof. *The distribution of these two experiments does not depend on the bit* b. □

By combining Claims 6–8 and the fact that $\mathbf{Validity}_1$ is satisfied, we conclude that

$$\mathbf{H}_0^{b,0} \approx_c \cdots \approx_c \mathbf{H}_0^{b,n} \equiv \mathbf{H}_1^{b,0} \approx_c \cdots \approx_c \mathbf{H}_1^{b,n} \equiv \mathbf{H}_1^{1-b,n}.$$

This concludes the proof. □

Lemma 5. *Let* $j_0, j_1 \in [n]$. *If* PE *is CPA-1-sided secure without collusions (Definition 9),* SKE *is CPA secure (Definition 4), and* LOBF *is secure (Definition 2), then*

$$\left| \mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{\text{0-CPA-1-iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_{2,j_0,j_1} \right] - \frac{1}{2} \right| \leq \mathit{negl}(\lambda).$$

Proof. *Without loss of generality, let* $q = |\mathcal{Q}_1| = \cdots = |\mathcal{Q}_n| \in \mathit{poly}(\lambda)$. *Consider the following hybrid experiments:*

- $\mathbf{H}_0^b(\lambda)$*: This is exactly the experiment* $\mathbf{G}_{\Pi,\mathsf{A}}^{\text{0-CPA-1-iPE}}(\lambda)$ *conditioned to the event* $\mathbf{Validity}_{2,j_0,j_1}$ *where the challenge bit is* b, *i.e., the adversary is valid and satisfies* $\mathbf{Validity}_{2,j_0,j_1}$.
- $\mathbf{H}_1^b(\lambda)$*: Same as* $\mathbf{H}_0^b$, *except that the challenger changes how it computes the challenger ciphertext* c_{j_b}. *Formally, the value* $c_{j_b}^{(1)}$ *of the challenge ciphertext* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}, c_{j_b}^{(2)})$ *is computed as* $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *(instead of* $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_{j_b}, \mathsf{k}_{j_b+1}))$*) where* $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$ *(for some function* k*),* $x_{j_b} = x_{j_b}^b$, *and* $x_j = x_j^\star$ *for* $j \in [n] \backslash \{j_b\}$. *Note that* $c_{j_b}^{(1)}$ *still depends on the challenge bit* b *since it is computed over the input* $(x_1, \ldots x_n)$ *where* $x_{j_b} = x_{j_b}^b$. *We will remove this dependency in* $\mathbf{H}_{5+n-1}^{b,0,0,0}$.[24]
- $\mathbf{H}_2^{b,0}$*: Identical to* $\mathbf{H}_1^b(\lambda)$.
- $\mathbf{H}_2^{b,i}(\lambda)$ *for* $i \in [q]$*: Same as* $\mathbf{H}_2^{b,i-1}(\lambda)$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_b}, \cdot, \cdot)$. *Formally, on input the* i'*th query* (x, m) *such that* $i' \leq i$, *the challenger computes* $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},$

[24] This allow us to reuse the proof in Lemmas 6 and 7.

$(x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{j_b} = x$, *and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{j_b\}$. *Finally, the challenger returns* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}, c_{j_b}^{(2)})$ *where* $c_{j_b}^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_{j_b}, c_{j_b}^{(1)})$, $y_{j_b} \leftarrow\$ \leftarrow\$^{s(\lambda)}$, *and* $\widetilde{\mathbb{C}}_{j_b} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_{j_b}, y_{j_b}, m)$. *Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$, *the challenger answers as usual, i.e., as defined in* $\mathbf{H}_2^{b,0}$.

$\mathbf{H}_3^b(\lambda)$: *Same as* $\mathbf{H}_2^{b,q}$, *except that the challenger changes how it computes the challenger ciphertext* c_{j_b}. *Formally, the value* $\widetilde{\mathbb{C}}_{j_b}$ *of challenge ciphertext* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}, c_{j_b}^{(2)})$ *is simulated by the challenger using the simulator of the lockable obfuscation scheme* LOBF, *i.e.,* $\widetilde{\mathbb{C}}_{j_b} \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_{j_b}|}, 1^{|m_{j_b}^b|})$.

$\mathbf{H}_4^{b,0}$: *Identical to* $\mathbf{H}_3^b(\lambda)$.

$\mathbf{H}_4^{b,i}(\lambda)$ *for* $i \in [q]$: *Same as* $\mathbf{H}_4^{b,i-1}(\lambda)$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_b}, \cdot, \cdot)$. *Formally, on input the* i'*th query* (x, m) *such that* $i' \leq i$, *the challenger returns* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}, c_{j_b}^{(2)})$ *where* $\widetilde{\mathbb{C}}_{j_b}$ *is computed using the simulator of the lockable obfuscator scheme* LOBF, *i.e.,* $\widetilde{\mathbb{C}}_{j_b} \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_{j_b}|}, 1^{|m|})$. *Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$, *the challenger answers as usual, i.e., as defined in* $\mathbf{H}_4^{b,0}$.

$\mathbf{H}_4^{b,q,q,1}$: *Identical to* $\mathbf{H}_4^{b,q}(\lambda)$.

$\mathbf{H}_{5+i}^{b,0,0,0}$ *for* $i \in \{0\} \cup [n-1]$: *Same as* $\mathbf{H}_{5+i-1}^{b,q,q,1}$ *except that the challenger changes how it computes the challenger ciphertext* c_v *where* $v = (j_b + i \mod n) + 1$. *Formally, the value* $c_v^{(1)}$ *is computed as* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$ *(for some function* k*),* $x_v = x_v^0$, *and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{v\}$. *Observe that* $c_v^{(1)}$ *is computed by fixing* $x_v = x_v^0$ *(instead of* $x_v = x_v^b$*), i.e., the predicate input* $(x_1, \ldots, x_n)$ *used to compute the* v*th challenge ciphertext is fixed and does not depend on the challenge bit* b.

$\mathbf{H}_{5+i}^{b,t_1,0,0}$ *for* $t_1 \in [q]$, $i \in \{0\} \cup [n-2]$: *Same as* $\mathbf{H}_{5+i}^{b,t_1-1,0,0}(\lambda)$ *except that the challenger changes how it answers to the first* t_1 *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_v, \cdot, \cdot)$ *where* $v = (j_b + i \mod n) + 1$. *On input the* t_1'*th query* (x, m) *such that* $t_1' \leq t_1$, *the challenger computes* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, *and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{v\}$. *Finally, the challenger returns* $c_v = (\widetilde{\mathbb{C}}_v, c_v^{(2)})$ *where* $c_v^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$, *and* $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)}$. *Otherwise, on input the* t_1'*th query* (x, m) *such that* $t_1' > t_1$, *the challenger answers as usual, i.e., as defined in* $\mathbf{H}_{5+i}^{b,0,0,0}$.

$\mathbf{H}_{5+i}^{b,q,t_2,0}$ *for* $t_2 \in [q]$, $i \in \{0\} \cup [n-2]$: *Same as* $\mathbf{H}_{5+i}^{b,q,t_2-1,0}(\lambda)$ *except that the challenger changes how it answers to the first* t_2 *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_v, \cdot, \cdot)$ *where* $v = (j_b + i \mod n) + 1$. *Formally, on input the* t_2'*th query* (x, m) *such that* $t_2' \leq t_2$, *the challenger returns* $c_v = (\widetilde{\mathbb{C}}_v, c_v^{(2)})$ *where* $\widetilde{\mathbb{C}}_v$ *is computed using the simulator of the lockable obfuscator scheme* LOBF, *i.e.,* $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathcal{V}_v|}, 1^{|m|})$. *Otherwise, on input the* t_2'*th query* (x, m) *such that* $t_2' > t_2$, *the challenger answers as usual, i.e., as defined in* $\mathbf{H}_{5+i}^{b,q,0,0}$.

$\mathbf{H}_{5+i}^{b,q,q,1}$ *for* $i \in \{0\} \cup [n-2]$*: Same as* $\mathbf{H}_{5+i}^{b,q,q,0}(\lambda)$ *except that the challenger computes the challenger ciphertext* c_v *differently for* $v = (j_b + i \mod n) + 1$*. Formally, the value* $\widetilde{\mathbb{C}}_v$ *of challenge ciphertext* $c_v = (\widetilde{\mathbb{C}}_v, c_v^{(2)})$ *is simulated by the challenger using the simulator of the lockable obfuscation scheme* LOBF*, i.e.,* $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_v|}, 1^{|m_v^b|})$.

Claim 9. $\mathbf{H}_0^b(\lambda) \approx_c \mathbf{H}_1^b(\lambda)$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_0^b(\lambda)$ *and* $\mathbf{H}_1^b(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA-1-sided security without collusions of* PE. A *is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ *for* $j \in [n]$*. Let* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, \mathsf{k}_{j+1})$ *for* $j \in [n]$ *where* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen*, forward the query* P^* *to* KGen_1 *and return the answer* dk_{P^*}.
   - *On input* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$*, return* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)}) \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
4. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D*. Send the challenge* $(m_*^0 = (y_{j_b}, \mathsf{k}_{j_b+1}), m_*^1 = 0^{s(\lambda)+k(\lambda)}, x_*^0 = (x_{*1}^0, \ldots, x_{*n}^0), x_*^1 = (x_{*1}^1, \ldots, x_{*n}^1))$ *where* $y_{j_b} \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$, $x_{*j_b}^0 = x_{*j_b}^1 = x_{j_b}^b$ *and* $x_{*j}^0 = x_{*j}^1 = x_j^\star$ *for* $j \in [n] \backslash \{j_b\}$.
5. *Receive the challenge ciphertext* c^* *from the challenger. Set* $c_{j_b}^{(1)} = c^*$.
6. *For any* $j \in [n] \backslash \{j_b\}$*, compute* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_j = x_j^b$*, and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n] \backslash \{j\}$.
7. *Compute* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *where* $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{ek}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *for any* $j \in [n]$.
8. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
9. *Answer to the incoming oracle queries as in Item* 3.
10. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D*. In particular, if* $d = 0$*,* A *simulates* $\mathbf{H}_0^b(\lambda)$*. On the other hand, if* $d = 1$*,* A *simulates* $\mathbf{H}_1^b(\lambda)$*. Moreover, since* D *submits a single query* P^* *to oracle* $\mathsf{KGen}(\mathsf{msk}, \cdot)$ *and it satisfies the condition* $\mathbf{Validity}_{2, j_0, j_1}$*, we know that* $P_{j_b}^*(x_{j_b}^b) = 0$*. Because of this,* A *submits only a query to oracle* $\mathsf{KGen}_1(\mathsf{msk}, \cdot)$ *(i.e., security without collusions) and, it is also a valid adversary for the experiment* $\mathbf{G}_{\mathsf{PE},\mathsf{A}}^{\mathsf{CPA\text{-}1\text{-}PE}}(\lambda)$ *with the same advantage of* D*. This concludes the proof.* □

Claim 10. $\mathbf{H}_2^{b,i-1}(\lambda) \approx_c \mathbf{H}_2^{b,i}(\lambda)$ *for* $i \in [q]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_2^{b,i-1}(\lambda)$ *and* $\mathbf{H}_2^{b,i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA-*1*-sided security without collusions of* PE. A *is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $\mathsf{k}_j \leftarrow\$ \mathsf{KGen}_2(1^\lambda)$ *for* $j \in [n]$. *Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *forward the query* P^* *to* KGen_1 *and return the answer* dk_{P^*}.
   - *On input* i'*th query* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$, A *proceeds as follows:*

     Case $j \neq j_b$: *Sample* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$. *Compute* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $x_j = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$.

     Case $j = j_b$ *and* $i' < i$: *Sample* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$. *Compute* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{j_b} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$.

     Case $j = j_b$ *and* $i' = i$: *Sample* $y_{j_b} \leftarrow\$ \leftarrow\$^{s(\lambda)}$ *and send* $(m_*^0 = (y_{j_b}, \mathsf{k}_{j_b+1}), m_*^1 = 0^{s(\lambda)+k(\lambda)}, x_*^0 = (x_{*1}^0, \ldots, x_{*n}^0), x_*^1 = (x_{*1}^1, \ldots, x_{*n}^1))$ *to the challenger where* $x^0_{*j_b} = x^1_{*j_b} = x$ *and* $x^0_{*j'} = x^1_{*j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$. *Receive the challenge ciphertext* c^* *and* $c_{j_b}^{(1)} = c^*$.

     Case $j = j_b$ *and* $i' > i$: *Sample* $y_{j_b} \leftarrow\$ \leftarrow\$^{s(\lambda)}$. *Compute* $c_{j_b}^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_{j_b}, \mathsf{k}_{j_b+1}))$ *where* $x_{j_b} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *where* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$.
4. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
5. *For every* $j \in [n] \setminus \{j_b\}$, *sample* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$ *and compute* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $x_j = x_j^b$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$.
6. *Sample* $y_{j_b} \leftarrow\$ \leftarrow\$^{s(\lambda)}$ *and compute the ciphertext* $c_{j_b}^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{j_b} = x_{j_b}^b$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$.
7. *Compute the ciphertext* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *where* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *for any* $j \in [n]$.
8. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
9. *Answer to the incoming oracle queries as in Item* 3.
10. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_2^{b,i-1}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_2^{b,i}(\lambda)$. *Moreover, since* D *submits a single query* P^* *to oracle* $\mathsf{KGen}(\mathsf{msk}, \cdot)$ *and it satisfies the condition* $\mathbf{Validity}_{2,j_0,j_1}$, *we know that* $\forall x'_{j_b} \in \mathcal{Q}_{j_b}$, $P^*_{j_b}(x'_{j_b}) = 0$. *Because of this,* A *submits a single query to oracle* $\mathsf{KGen}_1(\mathsf{msk}, \cdot)$

and it is also a valid adversary for the experiment $\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}PE}}_{\mathsf{PE},\mathsf{A}}(\lambda)$ *with the same advantage of* D. *This concludes the proof.* □

Claim 11. $\mathbf{H}^{b,q}_2(\lambda) \approx_c \mathbf{H}^b_3(\lambda)$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}^{b,q}_2(\lambda)$ *and* $\mathbf{H}^b_3(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, \mathsf{k}_{j+1})$ *for* $j \in [n]$. *Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, P^*)$.
   - *On input* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j = j_b$: *Sample* $y_{j_b} \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$. *Compute* $c^{(1)}_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{j_b} = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{j_b\}$.

     Case $j \neq j_b$: *Run* $c^{(1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{j\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c^{(2)}_j)$ *where* $c^{(2)}_j \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c^{(1)}_j)$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$.
3. *Receive the challenge* $((m^0_1, \ldots, m^0_n), (m^1_1, \ldots, m^1_n), (x^0_1, \ldots, x^0_n), (x^1_1, \ldots, x^1_n))$ *from* D.
4. *Compute* $c^{(1)}_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *and* $c^{(2)}_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c^{(1)}_{j_b})$ *where* $x_{j_b} = x^b_{j_b}$, *and* $x_j = x^\star_j$ *for* $j \in [n]\backslash\{j_b\}$.
5. *For any* $j \in [n] \setminus \{j_b\}$, *sample* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and compute* $c^{(1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$, $c^{(2)}_j \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c^{(1)}_j)$, *and* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m^b_j)$ *where* $x_j = x^b_j$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$.
6. *Send the challenge* $(\mathbb{V}_{j_b}, m^b_{j_b})$ *to the challenger and receive* $\widetilde{\mathbb{C}}$. *Set* $\widetilde{\mathbb{C}}_{j_b} = \widetilde{\mathbb{C}}$.
7. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c^{(2)}_j)$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
8. *Answer to the incoming oracle queries as in Item* 2.
9. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}^{b,q}_2(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}^b_3(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 12. $\mathbf{H}^{b,i-1}_4(\lambda) \approx_c \mathbf{H}^{b,i}_4(\lambda)$ *for* $i \in [q]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_4^{b,i-1}(\lambda)$ *and* $\mathbf{H}_4^{b,i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ *for* $j \in [n]$. *Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\$ \mathsf{KGen}(\mathsf{msk}, P^*)$.
   - *On input the* i'*th query* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j = j_b$ *and* $i' < i$: *Run* $\widetilde{\mathbb{C}}_{j_b} \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_{j_b}|}, 1^{|m|})$, $c_{j_b}^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_{j_b}, c_{j_b}^{(1)})$, *and* $c_{j_b}^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{j_b} = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n] \backslash \{j_b\}$.

     Case $j = j_b$ *and* $i' = i$: *Compute* $c_{j_b}^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_{j_b}, c_{j_b}^{(1)})$ *and* $c_{j_b}^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{j_b} = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n] \backslash \{j_b\}$. *Send the challenge* $(\mathbb{V}_{j_b}, m)$ *to the challenger and receive the answer* $\widetilde{\mathbb{C}}^*$. *Set* $\widetilde{\mathbb{C}}_{j_b} = \widetilde{\mathbb{C}}^*$.

     Case $j = j_b$ *and* $i' > i$: *Compute* $\widetilde{\mathbb{C}}_{j_b} \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_{j_b}, m)$, $c_{j_b}^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_{j_b}, c_{j_b}^{(1)})$, *and* $c_{j_b}^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $y_{j_b} \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_{j_b} = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n] \backslash \{j_b\}$.

     Case $j \neq j_b$: *Compute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n] \backslash \{j\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$.
3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *Run* $\widetilde{\mathbb{C}}_{j_b} \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_{j_b}|}, 1^{|m_{j_b}^b|})$, $c_{j_b}^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$, *and* $c_{j_b}^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_{j_b}^{(1)})$ *where* $x_{j_b} = x_{j_b}^b$, *and* $x_j = x_j^\star$ *for* $j \in [n] \backslash \{j_b\}$.
5. *For any* $j \in [n] \setminus \{j_b\}$, *sample* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$ *and compute* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *where* $x_j = x_j^b$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n] \backslash \{j\}$.
6. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
7. *Answer to the incoming oracle queries as in Item* 2.
8. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_4^{b,i-1}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_4^{b,i}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 13. $\mathbf{H}^{b,q,q,1}_{5+i-1}(\lambda) \approx_c \mathbf{H}^{b,0,0,0}_{5+i}(\lambda)$ *for* $i \in \{0\} \cup [n-1]$.

Proof. *Let* $v = (j_b + i \mod n) + 1$. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}^{b,q,q,1}_{5+i-1}(\lambda)$ *and* $\mathbf{H}^{b,0,0,0}_{5+i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA security of* SKE. A *is defined as follows:*

1. *Compute* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$ *and* $\mathsf{ek}_j = (\mathsf{ek}, \mathsf{k}_j, \mathsf{k}_{j-1})$ *for* $j \in [n]\backslash\{v\}$. *If* $v \neq 1$, *let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
   - *On input* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j \in [j_b : v-1]^+_n$: *Compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m|})$, $c^{(2)}_j \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c^{(1)}_j)$, *and* $c^{(1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{j\}$.

     Case $j = v$: *Run* $c^{(1)}_v \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_v, \mathsf{k}_{v+1}))$ *where* $y_v \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_v = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{v\}$. *Send the query* $c^{(1)}_v$ *to the oracle* Enc_2 *and receive the answer* $c^{(2)}_v$. *Compute* $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$.

     Case $i < n-2$ *(hence,* $v \notin \{j_b - 1, j_b\}$*) and* $j \in [v+1 : j_b - 1]^+_n$: *Run* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$, *the ciphertext* $c^{(2)}_j \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c^{(1)}_j)$, *and* $c^{(1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{j\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c^{(2)}_j)$.
3. *Receive the challenge* $((m^0_1, \ldots, m^0_n), (m^1_1, \ldots, m^1_n), (x^0_1, \ldots, x^0_n), (x^1_1, \ldots, x^1_n))$ *from* D.
4. **Case** $i < n-1$ **(hence,** $v \neq j_b$**):** *For every* $j \in [n]$, *the adversary* A *proceeds as follows:*

   Case $j \in [j_b, v-1]^+_n$: *Run* $c^{(1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x^b_j$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m^b_j|})$ *and* $c^{(2)}_j \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c^{(1)}_j)$.

   Case $j = v$: *Run* $c^{(1,0)}_v \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x^0_{*1}, \ldots, x^0_{*n}), (y_v, \mathsf{k}_{v+1}))$ *and* $c^{(1,1)}_v \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x^1_{*1}, \ldots, x^1_{*n}), 0^{s(\lambda)+k(\lambda)})$ *where* $y_v \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x^0_{*v} = x^b_v$, $x^1_{*v} = x^0_v$, *and* $x^0_{*j'} = x^1_{*j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{v\}$. *Send the challenge* $(m^0 = c^{(1,0)}_v, m^1 = c^{(1,1)}_v)$ *to the challenger and receive the answer* c^*. *Set* $c^{(2)}_v$ *and compute* $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, m^b_v)$.

   Case $i < n-2$ *(hence,* $v \notin \{j_b - 1, j_b\}$*) and* $j \in [v+1 : j_b - 1]^+_n$: *Run* $c^{(1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_j = x^b_j$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m^b_j)$ *and* $c^{(2)}_j \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c^{(1)}_j)$.

5. **Otherwise, case $i = n-1$ (hence, $v = j_b$):** *For every $j \in [n]$, the adversary* A *proceeds as follows:*

   Case $j \in [j_b+1 : j_b-1]_n^+$: Execute $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^b$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^b|})$ *and* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

   Case $j = j_b$: Run $c_{j_b}^{(1,0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_{*1}^0, \ldots, x_{*n}^0), 0^{s(\lambda)+k(\lambda)})$ *and* $c_{j_b}^{(1,1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_{*1}^1, \ldots, x_{*n}^1), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{*j_b}^0 = x_{j_b}^b$, $x_{*j_b}^1 = x_{j_b}^0$, *and* $x_{*j'}^0 = x_{*j'}^1 = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j_b\}$. *Send the challenge* $(m^0 = c_{j_b}^{(1,0)}, m^1 = c_{j_b}^{(1,1)})$ *to the challenger and receive the answer* c^*. *Set* $c_{j_b}^{(2)}$. *Finally, compute* $\widetilde{\mathbb{C}}_{j_b} \leftarrow\$ \widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_{j_b}|}, 1^{|m_{j_b}^b|})$.

6. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
7. *Answer to the incoming oracle queries as in* 2.
8. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if $d = 0$,* A *simulates* $\mathbf{H}_{5+i-1}^{b,q,q,1}(\lambda)$. *On the other hand, if $d = 1$,* A *simulates* $\mathbf{H}_{5+i}^{b,0,0,0}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 14. $\mathbf{H}_{5+i}^{b,t_1-1,0,0}(\lambda) \approx_c \mathbf{H}_{5+i}^{b,t_1,0,0}(\lambda)$ *for $t_1 \in [q]$ and $i \in \{0\} \cup [n-2]$.*

Proof. *Let* $v = (j_b + i \mod n) + 1$ *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_{5+i}^{b,t_1-1,0,0}(\lambda)$ *and* $\mathbf{H}_{5+i}^{b,t_1,0,0}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA security of* SKE. A *is defined as follows:*

1. *Compute* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}_1(1^\lambda)$ *and* $\mathsf{ek}_j = (\mathsf{ek}, \mathsf{k}_j, \mathsf{k}_{j-1})$ *for* $j \in [n]\backslash\{v\}$. *If* $v \neq 1$, *let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\$ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
   - *On input the* t_1'*th query* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j \in [j_b : v-1]_n^+$: *Execute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(, \mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{j\}$.

     Case $j = v$ and $t_1' < t_1$: Sample $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)}$. *Run* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{v\}$. *Send the query* $c_v^{(1)}$ *to the oracle* Enc_2 *and receive the answer* $c_v^{(2)}$. *Compute* $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$.

     Case $j = v$ and $t_1' = t_1$: Compute $c_v^{(1,0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_v, \mathsf{k}_{v+1}))$ *and* $c_v^{(1,1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* y_v

$\leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_v = x$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{v\}$. *Send the challenge* $(m^0 = c_v^{(1,0)}, m^1 = c_v^{(1,1)})$ *to the challenger and receive the answer* c^*. *Set* $c_v^{(2)}$. *Finally, compute* $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$.

Case $j = v$ *and* $t'_1 > t_1$: *Sample* $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)}$. *Run* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (k_v, \mathsf{k}_{v+1}))$ *where* $x_v = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{v\}$. *Send the query* $c_v^{(1)}$ *to the oracle* Enc_2 *and receive the answer* $c_v^{(2)}$. *Compute* $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$.

Case $i < n-2$ *(hence,* $v \neq j_b - 1$*) and* $j \in [v+1 : j_b - 1]_n^+$: *Run* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{j\}$.

Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$.

3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$, *the adversary* A *proceeds as follows:*

Case $j \in [j_b : v-1]_n^+$: *Run* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^b|})$ *and* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

Case $j = v$: *Sample* $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)+k(\lambda)}$ *and compute* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_v = x_v^0$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{v\}$. *Send the query* $c_v^{(1)}$ *to the oracle* Enc_2 *and receive the answer* $c_v^{(2)}$. *Compute* $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$.

Case $i < n-2$ *(hence,* $v \neq j_b - 1$*) and* $j \in [v+1 : j_b - 1]_n^+$: *Run* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_j = x_j^b$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *and* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

5. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
6. *Answer to the incoming oracle queries as in Item* 2.
7. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_{5+i}^{b,t_1-1,0,0}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_{5+i}^{b,t_1,0,0}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 15. $\mathbf{H}_{5+i}^{b,q,t_2-1,0}(\lambda) \approx_c \mathbf{H}_{5+i}^{b,q,t_2,0}(\lambda)$ *for* $t_2 \in [q]$ *and* $i \in \{0\} \cup [n-2]$.

Proof. *Let* $v = (j_b + i \mod n) + 1$. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_{5+i}^{b,q,t_2-1,0}(\lambda)$ *and* $\mathbf{H}_{5+i}^{b,q,t_2,0}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscator scheme* LOBF. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, k_{j+1})$ *for* $j \in [n]$. *Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\$ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
   - *On input the* t_2'*th query* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j \in [j_b : v-1]_n^+$*: Execute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{j\}$.

     Case $j = v$ *and* $t_2' < t_2$*: Run* $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_v|}, 1^{|m|})$, $c_v^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{v\}$.

     Case $j = v$ *and* $t_2' = t_2$*: Compute* $c_v^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$ *and* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{v\}$. *Send the challenge* $(\mathbb{V}_v, m)$ *to the challenger and receive the answer* $\widetilde{\mathbb{C}}^*$. *Set* $\widetilde{\mathbb{C}}_v = \widetilde{\mathbb{C}}^*$.

     Case $j = v$ *and* $t_2' > t_2$*: Sample* $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)}$. *Compute* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{v\}$. *Send the query* $c_v^{(1)}$ *to the oracle* Enc_2 *and receive the answer* $c_v^{(2)}$. *Compute* $\widetilde{\mathbb{C}}_v \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, y_v, m)$.

     Case $i < n-2$ *(hence,* $v \neq j_b - 1$*) and* $j \in [v+1 : j_b - 1]_n^+$*: Run* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{j\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$.
3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$, *the adversary* A *proceeds as follows:*

   Case $j \in [j_b : v-1]_n^+$*: Run* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^b|})$ *and* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

   Case $j = v$*: Sample* $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)+k(\lambda)}$ *and compute* $c_v^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $y_v \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_v = x_v^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in$

$[n]\backslash\{v\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m_v^b)$ *and* $c_v^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$.

Case $i < n-2$ *(hence,* $v \neq j_b - 1$*) and* $j \in [v+1 : j_b - 1]_n^+$*: Run* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_j = x_j^b$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *and* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

5. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
6. *Answer to the incoming oracle queries as in Item* 2.
7. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_{5+i}^{b,q,t_2-1,0}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_{5+i}^{b,q,t_2,0}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 16. $\mathbf{H}_{5+i}^{b,q,q,0}(\lambda) \approx_c \mathbf{H}_{5+i}^{b,q,q,1}(\lambda)$ *for* $i \in \{0\} \cup [n-2]$.

Proof. *Let* $v = (j_b + i \mod n) + 1$. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_{5+i}^{b,q,q,0}(\lambda)$ *and* $\mathbf{H}_{5+i}^{b,q,q,1}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscator scheme* LOBF. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, \mathsf{k}_{j+1})$ *for* $j \in [n]$. *Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\$ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
   - *On input* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j \in [j_b : v]_n^+$*: Run* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{j\}$.

     Case $i < n-2$ *(hence,* $v \neq j_b - 1$*) and* $j \in [v+1 : j_b - 1]_n^+$*: Run* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$, $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{j\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$.
3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$, *the adversary* A *proceeds as follows:*

Case $j \in [j_b : v-1]_n^+$*: Run* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$*, and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$*. Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^b|})$ *and* $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

Case $j = v$*: Run* $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$ *and* $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x_v^0$ *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{v\}$*. Send the challenge* $(\mathbb{V}_v, m_v^b)$ *to the challenger and receive the answer* $\widetilde{\mathbb{C}}^*$*. Set* $\widetilde{\mathbb{C}}_v = \widetilde{\mathbb{C}}^*$.

Case $i < n-2$ *(hence,* $v \neq j_b - 1$*) and* $j \in [v+1 : j_b - 1]_n^+$*: Run* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{\,s(\lambda)}$*,* $x_j = x_j^b$*, and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$*. Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *and* $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

5. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
6. *Answer to the incoming oracle queries as in Item* 2.
7. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D*. In particular, if* $d = 0$*,* A *simulates* $\mathbf{H}_{5+i}^{b,q,q,0}(\lambda)$*. On the other hand, if* $d = 1$*,* A *simulates* $\mathbf{H}_{5+i}^{b,q,q,1}(\lambda)$*. Hence,* A *has the same advantage of* D*. This concludes the proof.* □

Claim 17. $\mathbf{H}_{5+n}^{1-b,q,q,q}(\lambda) \equiv \mathbf{H}_{5+n}^{b,q,q,1}(\lambda)$.

Proof. *The distributions of these two experiments do not depend on the bit* b. □

By combining Claims 9–17 and the fact that $\mathbf{Validity}_{2,j_0,j_1}$ holds, we conclude that

$$\begin{aligned}\mathbf{H}_0^b \approx_c \mathbf{H}_1^b \equiv \mathbf{H}_2^{b,0} \approx_c \cdots \approx_c \mathbf{H}_2^{b,q} \approx_c \mathbf{H}_3^b \equiv \mathbf{H}_4^{b,0} \approx_c \cdots \approx_c \mathbf{H}_4^{b,q} \equiv \\ \mathbf{H}_4^{b,q,q,1} \approx_c \mathbf{H}_5^{b,0,0,0} \approx_c \cdots \approx_c \mathbf{H}_5^{b,q,0,0} \approx_c \cdots \approx_c \mathbf{H}_5^{b,q,q,0} \approx_c \\ \mathbf{H}_5^{b,q,q,1} \approx_c \cdots \approx_c \mathbf{H}_{5+n-1}^{b,0,0,0} \equiv \mathbf{H}_{5+n-1}^{1-b,0,0,0}.\end{aligned}$$

This concludes the proof. □

Lemma 6. *Let* $j_0 \in [n]$*. If* PE *is CPA-1-sided secure without collusions (Definition 9),* SKE *is CPA secure (Definition 4), and* LOBF *is secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{0\text{-}\mathsf{CPA}\text{-}1\text{-}\mathsf{iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_{3,j_0}\right] - \frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *Without loss of generality, let* $q = |\mathcal{Q}_1| = \ldots = |\mathcal{Q}_{nim}| \in \mathit{poly}(\lambda)$*. Consider the hybrid experiments of Lemmas 4 and 5. Formally,*

- *Let* $\mathbf{H}_0^{1,i}(\lambda)$ *and* $\mathbf{H}_1^{1,i}(\lambda)$ *for* $i \in \{0\} \cup [n]$ *be the hybrids of Lemma 4 (for the challenge bit* $b = 1$*) except that are conditioned to the event* $\mathbf{Validity}_{3,j_0}$ *(instead of* $\mathbf{Validity}_1$*).*
- *Let* $\mathbf{H}_0^0(\lambda), \mathbf{H}_1^0(\lambda), \mathbf{H}_2^{0,i}(\lambda), \mathbf{H}_3^0(\lambda), \mathbf{H}_4^{0,i}(\lambda), \mathbf{H}_4^{0,q,q,1}(\lambda), \mathbf{H}_{5+j}^{0,i,0,0}(\lambda), \mathbf{H}_{5+j}^{0,q,i,0}(\lambda), \mathbf{H}_{5+j}^{0,q,q,k}(\lambda)$, *and* $\mathbf{H}_{5+n-1}^{0,0,0,0}(\lambda)$*, for* $(i, j, k) \in (\{0\} \cup [q]) \times (\{0\} \cup [n-2]) \times \leftarrow\!\!\$$*, be the hybrids of Lemma 5 (for the challenge bit* $b = 0$*) except that are conditioned to the event* $\mathbf{Validity}_{3,j_0}$ *(instead of* $\mathbf{Validity}_{2,j_0,j_1}$*).*

In addition, consider the following additional hybrids experiments:

$\mathbf{H}_{5+n}^{0,q,q}$*: Identical to* $\mathbf{H}_{5+n-1}^{0,0,0,0}$.

$\mathbf{H}_{5+n+i}^{0,0,0}$ *for* $i \in [n]$*: Identical to* $\mathbf{H}_{5+n+i-1}^{0,q,q}$.

$\mathbf{H}_{5+n+i}^{0,0,t_2}$ *for* $t_2 \in [q], i \in [n]$*: Same as* $\mathbf{H}_{5+n+i}^{0,0,t_2-1}$ *except that the challenger changes how it answers to the first* t_2 *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_v, \cdot, \cdot)$ *where* $v = (j_0 - i - 1 \mod n) + 1$*. Formally, on input the* t_2'*th query* (x, m) *such that* $t_2' \le t_2$*, the challenger returns* $c_v = (\widetilde{\mathbb{C}}_v, c_v^{(2)})$ *where* $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$ *where* $y_v \leftarrow\!\!\$ \leftarrow\!\!\$^{s(\lambda)}$*. Otherwise, on input the* t_2'*th query* (x, m) *such that* $t_2' > t_2$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_{5+n+i}^{0,0,0}$.

$\mathbf{H}_{5+n+i}^{0,t_1,q}$ *for* $t_1 \in [q], i \in [n]$*: Same as* $\mathbf{H}_{5+n+i}^{0,t_1-1,q}$ *except that the challenger changes how it answers to the first* t_1 *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_v, \cdot, \cdot)$ *where* $v = (j_0 - i - 1 \mod n) + 1$*. Formally, on input the* t_1'*th query* (x, m) *such that* $t_1' \le t_1$*, the challenger computes* $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_v, \mathsf{k}_{v+1}))$ *where* $y_v \leftarrow\!\!\$ \leftarrow\!\!\$^{s(\lambda)}$*,* $x_v = x$*, and* $x_j = x_j^\star$ *for* $j \in [n] \backslash \{v\}$*. Finally, the challenger returns* $c_v = (\widetilde{\mathbb{C}}_v, c_v^{(2)})$ *where* $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$*,* $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$*. Otherwise, on input the* t_1'*th query* (x, m) *such that* $t_1' > t_1$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_{5+n+i}^{0,0,q}$.

Claim 18. $\mathbf{H}_0^0(\lambda) \approx_c \mathbf{H}_{5+n-1}^{0,0,0,0}(\lambda)$.

Proof. *The proof of Claim 18 is identical to that of Lemma 5 where the challenge bit is* $b = 0$. □

Claim 19. $\mathbf{H}_{5+n+i}^{0,0,t_2-1}(\lambda) \approx_c \mathbf{H}_{5+n+i}^{0,0,t_2}(\lambda)$ *for* $t_2 \in [q]$ *and* $i \in [n]$.

Proof. *Let* $v = (j_0 - i - 1 \mod n) + 1$*. Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_{5+n+i}^{0,0,t_2-1}(\lambda)$ *and* $\mathbf{H}_{5+n+i}^{0,0,t_2}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscator scheme* LOBF*.* A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_1, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, k_{j+1})$ *for* $j \in [n]$*. Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen*, return* $\mathsf{dk}_{P^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
   - *On input the* t_2'*th query* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$*,* A *proceeds as follows:*

Case $i > 1$ and $j \in [j_0 - 1 : v + 1]_n^-$: Compute $\widetilde{\mathbb{C}}_j \leftarrow_\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$, $c_j^{(2)} \leftarrow_\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow_\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow_\$ \leftarrow_\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{j\}$.

Case $j = v$ and $t_2' < t_2$: Compute $\widetilde{\mathbb{C}}_j \leftarrow_\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$, $c_v^{(2)} \leftarrow_\$ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, *and* $c_v^{(1)} \leftarrow_\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $y_v \leftarrow_\$ \leftarrow_\$^{s(\lambda)}$, $x_v = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{v\}$.

Case $j = v$ and $t_2' = t_2$: Compute $c_v^{(2)} \leftarrow_\$ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, *and* $c_v^{(1)} \leftarrow_\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{v\}$. *Send the challenge* $(\mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, m)$ *to the challenger and receive* $\widetilde{\mathbb{C}}^*$. *Set* $\widetilde{\mathbb{C}}_v = \widetilde{\mathbb{C}}^*$.

Case $j = v$ and $t_2' > t_2$: Run $\widetilde{\mathbb{C}}_v \leftarrow_\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_v|}, 1^{|m|})$, $c_v^{(2)} \leftarrow_\$ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, $c_v^{(1)} \leftarrow_\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{v\}$.

Case $i \neq n$ and $j \in [v - 1 : j_0]_n^-$: Compute $\widetilde{\mathbb{C}}_j \leftarrow_\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m|})$, $c_j^{(2)} \leftarrow_\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow_\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x$, $x_{j'} = x^\star_{j'}$ *for any* $j' \in [n]\backslash\{j\}$.

Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$.

3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$, *the adversary* A *computes* $c_j^{(1)} \leftarrow_\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow_\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^0|})$ *and* $c_j^{(2)} \leftarrow_\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.
5. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
6. *Answer to the incoming oracle queries as in Item* 2.
7. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_{5+n+i}^{0,0,t_2}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_{5+n+i}^{0,0,t_2-1}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 20. $\mathbf{H}_{5+n+i}^{0,t_1-1,q}(\lambda) \approx_c \mathbf{H}_{5+n+i}^{0,t_1,q}(\lambda)$ *for* $t_1 \in [q]$ *and* $i \in [n-1]$.

Proof. *Let* $v = (j_0 - i - 1 \mod n) + 1$. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_{5+n+i}^{0,t_1,q}(\lambda)$ *and* $\mathbf{H}_{5+n+i}^{0,t_1-1,q}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA security of* SKE. A *is defined as follows:*

1. *Compute* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow_\$ \mathsf{Setup}_1(1^\lambda)$ *and* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ *for* $j \in [n]\backslash\{v\}$. *If* $v \neq 1$, *let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*

- *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
- *On input the* t_1'*th query* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

  Case $i > 1$ *and* $j \in [j_0 - 1 : v + 1]_n^-$*: Run* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_j, m)$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{j\}$.

  Case $j = v$ *and* $t_1' < t_1$*: Run* $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_v, \mathsf{k}_{v+1}))$ *where* $y_v \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_v = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{v\}$. *Send the query* $c_v^{(1)}$ *to the oracle* Enc_2 *and receive the answer* $c_v^{(2)}$. *Compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$.

  Case $j = v$ *and* $t_1' = t_1$*: Run* $c_v^{(1,0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *and* $c_v^{(1,1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_v, \mathsf{k}_{v+1}))$ *where* $y_v \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_v = x$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{v\}$. *Send the challenge* $(m^0 = c_v^{(1,0)}, m^1 = c_v^{(1,1)})$ *to the challenger and receive the answer* c^*. *Set* $c_v^{(2)}$ *and compute* $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$.

  Case $j = v$ *and* $t_1' > t_1$*: Run* $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{v\}$. *Send the query* $c_v^{(1)}$ *to the oracle* Enc_2 *and receive the answer* $c_v^{(2)}$. *Compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_v, y_v, m)$ *where* $y_v \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$.

  Case $j \in [v - 1 : j_0]_n^-$*: Run* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_v|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, *and* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{j\}$.

  Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$.

3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$, *the adversary* A *proceeds as follows:*

   Case $j = v$*: Compute* $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_v = x_v^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{v\}$. *Send the query* $c_v^{(1)}$ *to the oracle* Enc_2 *and receive the answer* $c_v^{(2)}$. *Finally, compute* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^0|})$.

   Case $j \neq v$*: Compute* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n] \setminus \{j\}$. *Finally, run* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^0|})$ *and* $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

5. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
6. *Answer to the incoming oracle queries as in Item* 2.
7. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}^{0,t_1-1,q}_{5+n+i}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}^{0,t_1,q}_{5+n+i}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 21. $\mathbf{H}^{0,t_1-1,q}_{5+2n}(\lambda) \approx_c \mathbf{H}^{0,t_1,q}_{5+2n}(\lambda)$ *for* $t_1 \in [q]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}^{0,t_1,q}_{5+2n}(\lambda)$ *and* $\mathbf{H}^{0,t_1-1,q}_{5+2n}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA-*1*-sided security without collusions of* PE. A *is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ *for* $j \in [n]$. *Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *forward the query* P^* *to* KGen_1 *and return the answer* dk_{P^*}.
   - *On input* t_1'*th query* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$, A *proceeds as follows:*

     Case $j \neq j_0$*: Sample* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and compute* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $x_j = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$.

     Case $j = j_0$ *and* $t_1' < t_1$*: Compute* $c_{j_0}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_{j_0}, \mathsf{k}_{j_0+1}))$ *where* $x_{j_0} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_0\}$.

     Case $j = j_0$ *and* $t_1' = t_1$*: Sample* $y_{j_0} \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and send the challenge* $(m^0_* = 0^{s(\lambda)+k(\lambda)}, m^1_* = (y_{j_0}, \mathsf{k}_{j_0+1}), x^0_* = (x^0_{*1}, \ldots, x^0_{*n}), x^1_* = (x^1_{*1}, \ldots, x^1_{*n}))$ *to the challenger where* $x^0_{*j_0} = x^1_{*j_0} = x$ *and* $x^0_{*j'} = x^1_{*j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_0\}$. *Receive the challenge ciphertext* c^* *and* $c_{j_0}^{(1)} = c^*$.

     Case $j = j_0$ *and* $t_1' > t_1$*: Sample* $y_{j_0} \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and compute* $c_{j_0}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_{j_0} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_0\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *where* $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m)$.
4. *Receive the challenge* $((m^0_1, \ldots, m^0_n), (m^1_1, \ldots, m^1_n), (x^0_1, \ldots, x^0_n), (x^1_1, \ldots, x^1_n))$ *from* D.
5. *For every* $j \in [n]$, *the adversary* A *computes* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x^0_j$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$.
6. *For every* $j \in [n]$, *the adversary* A *computes* $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{ek}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m^0_j|})$.
7. *Set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
8. *Answer to the incoming oracle queries as in Item* 3.

9. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A perfectly simulates the view of D. In particular, if $d = 0$, A simulates $\mathbf{H}_{5+2n}^{0,t_1-1,q}(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}_{5+2n}^{0,t_1,q}(\lambda)$. Moreover, since D submits a single query $\mathcal{P}^*$ to oracle $\mathsf{KGen}(\mathsf{msk},\cdot)$ and it satisfies $\mathbf{Validity}_{3,j_0}$, we know that $\forall x'_{j_0} \in \mathcal{Q}_{j_0}, P^*_{j_0}(x'_{j_0}) = 0$. Because of this, A submits a single query to oracle $\mathsf{KGen}_1(\mathsf{msk},\cdot)$ and it is also a valid adversary for the experiment $\mathbf{G}_{\mathsf{PE},\mathsf{A}}^{\mathsf{CPA\text{-}1\text{-}PE}}(\lambda)$ with the same advantage of D. This concludes the proof. □

Claim 22. $\mathbf{H}_0^{1,0}(\lambda) \approx_c \mathbf{H}_1^{1,q}(\lambda)$.

Proof. *The proof of Claim 22 is identical to that of Lemma 4 where the challenge bit is $b = 1$.* □

Claim 23. $\mathbf{H}_{5+2n}^{0,q,q}(\lambda) \equiv \mathbf{H}_1^{1,q}(\lambda)$.

Proof. *Claim 23 follows by observing that experiments $\mathbf{H}_{5+2n}^{0,q,q}(\lambda)$ and $\mathbf{H}_1^{1,q}(\lambda)$ are identical (and does not depend on the bit b).* □

By combining Claims 18–23 and the fact that $\mathbf{Validity}_{3,j_0,}$ is satisfied, we conclude that

$$\mathbf{H}_0^0 \approx_c \mathbf{H}_{5+n-1}^{0,0,0,0} \equiv \mathbf{H}_{5+n}^{0,q,q} \equiv \mathbf{H}_{5+n+1}^{0,0,0} \approx_c \ldots \approx_c \mathbf{H}_{5+n+1}^{0,0,q} \approx_c \ldots \approx_c \mathbf{H}_{5+n+1}^{0,q,q} \equiv \mathbf{H}_{5+n+2}^{0,0,0} \approx_c \ldots \approx_c \mathbf{H}_{5+2n}^{0,0,q} \approx_c \ldots \approx_c \mathbf{H}_{5+2n}^{0,q,q} \equiv \mathbf{H}_1^{1,q} \approx_c \mathbf{H}_0^{1,0}$$

This concludes the proof. □

Lemma 7. *Let $j_1 \in [n]$. If PE is CPA-1-sided secure without collusions (Definition 9), SKE is CPA secure (Definition 4), and LOBF is secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{0\text{-}\mathsf{CPA\text{-}1\text{-}iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_{4,j_1}\right] - \frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *Lemma 7 follows by using a symmetrical argument to that of Lemma 6.* □

By combining Lemmas 4–7, we conclude that Π is CPA-1-sided secure without collusions.

CPA-2-sided security of Π for $n = O(\log(\lambda))$ (Theorem 6). As usual, consider the predicate space $\mathcal{P} = \{P(x_1,\ldots,x_n)\}$ of Construction 3 where $P(x_1,\ldots,x_n) = P_1(x_1) \wedge \ldots \wedge P_n(x_n)$. Let $P^* \in \mathcal{P}$ be the only predicate for which the adversary will ask for the decryption key dk_{P^*} during the experiment $\mathbf{G}_{\Pi,\mathsf{A}}^{0\text{-}\mathsf{CPA\text{-}2\text{-}iPE}}$ (recall that we prove the security of Construction 3 in the scenario without collusions, i.e., $|\mathcal{Q}_{\mathsf{KGen}}| = 1$). Also, consider the validity condition of $\mathbf{G}_{\Pi,\mathsf{A}}^{0\text{-}\mathsf{CPA\text{-}2\text{-}iPE}}$ and consider the following observations:

1. Suppose that $\forall j \in [n], \forall i_1 \in [k_1+1], \ldots, \forall i_n \in [k_n+1]$, we have

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

where $\mathcal{Q}_i^b = \{x_i^{(1,b)}, \ldots, x_i^{(k_i,b)}, x_i^{(k_i+1,b)} = x_i^b\}$ for $i \in [n]$, $b \in \leftarrow\!\!\$$ as defined in Definition 13. This means that the adversary cannot decrypt any part of the challenge ciphertext.
2. Otherwise, if $\exists j \in [n], \exists i_1 \in [k_1+1], \ldots, \exists i_n \in [k_n+1]$ such that

$$P^*(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) = P^*(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) = 1, \tag{7}$$

we are guaranteed that the adversary can retrieve the message m_j^b contained into the jth challenge ciphertext c_j. By taking into account the definition of $P^*(x_1, \ldots, x_n) = P_1^*(x_1) \wedge \ldots \wedge P_n^*(x_n)$, Eq. (7) implies that, for any $j' \in [n] \backslash [j]$, the adversary can satisfy the ith predicate P_i^* for $i \in [n] \backslash [j']$ (e.g., by taking the ciphertexts corresponding to the indexes $i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_n$ and the jth challenge ciphertext c_j). Hence, the secrecy of the challenge message $m_{j'}^b$ solely depends on the evaluation of $P_{j'}^*$ over the challenge input $x_{j'}^b$.

By taking into account the following observations, we can rewrite the validity condition of $\mathbf{G}_{\Pi,\mathsf{A}}^{0\text{-CPA-2-iPE}}$ (Definition 13) in the following way:

$$\text{Either } \mathbf{Validity_1} \text{ or } \mathbf{Validity_2}$$

where $\mathbf{Validity}_1$ and $\mathbf{Validity}_2$ formalize the observations of Items 1 and 2 respectively, i.e.,

$$\begin{aligned}
\mathbf{Validity}_1 :\ & \forall j \in [n], \forall i_1 \in [k_1+1], \ldots, \forall i_n \in [k_n+1], \\
& P^*(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) = \\
& P^*(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) = 0 \\
\mathbf{Validity}_2 :\ & \forall j \in [n], \text{Either } P_j^*(x_j^0) = P_j^*(x_j^1) = 0 \text{ or } P_j^*(x_j^0) = P_j^*(x_j^1) \wedge m_j^0 = m_j^1
\end{aligned}$$

where $\mathcal{Q}_i^b = \{x_i^{(1,b)}, \ldots, x_i^{(k_i,b)}, x_i^{(k_i+1,b)} = x_i^b\}$ for $i \in [n]$, $b \in \leftarrow\!\!\$$ as defined in Definition 13. Hence, the CPA-2-sided security of Construction 3 follows by proving the following lemmas.

Lemma 8. *If* PE *is CPA-1-sided secure without collusions (Definition 9),* SKE *is CPA secure (Definition 4), and* LOBF *is secure (Definition 2), then*

$$\left| \mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{0\text{-CPA-2-iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_1 \right] - \frac{1}{2} \right| \leq \mathit{negl}(\lambda).$$

Proof. *Note that* $\mathbf{Validity}_1$ *is equivalent to the validity condition of CPA-1-sided security. Hence, the lemma follows by leveraging an identical argument to that of the CPA-1-sided case (Sect. 5.2.1).* □

Lemma 9. *If* PE *is CPA-2-sided secure without collusions (Definition 9) and* LOBF *is secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}^{\text{0-CPA-2-iPE}}_{\Pi,\mathsf{A}}(\lambda)=1 \wedge |\mathcal{Q}_{\mathsf{KGen}}|=1 \middle| \mathbf{Validity}_2\right]-\frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *In this lemma, we restrict the adversary to submit the (single) query to* KGen *only before the challenge phase, i.e., the oracle* KGen *is not available after the challenge phase. Under this restriction, we prove Lemma 9 for any* $n = \mathit{poly}(\lambda)$*. Then, we use complexity leveraging to show that the lemma holds when* $n = O(\log(\lambda))$ *and the oracle* KGen *is available after the challenge phase. Without loss of generality, we assume the adversary always submit a query to* KGen*. Finally, for the sake of clarity, in the rest of this proof we use the notation* $\mathbb{V}_i \stackrel{def}{=} \mathbb{C}_{c_i^{(2)},\mathsf{k}_{i+1}}$ *where* $c_i^{(2)}$ *and* k_{i+1} *will be clear from the context.*

Consider the following hybrid experiments:

$\mathbf{H}_0^{b,0}(\lambda)$*: This is exactly the experiment* $\mathbf{G}^{\text{0-CPA-2-iPE}}_{\Pi,\mathsf{A}}(\lambda)$ *conditioned to the event* $\mathbf{Validity}_2$ *where the challenge bit is b, i.e., the adversary is valid and satisfied* $\mathbf{Validity}_2$*. Recall that the oracle* KGen *is not available after the challenge phase.*

$\mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}_0^{b,i-1}$*, except that the challenger changes how it computes the challenger ciphertext* c_i*. Let* $P^* \in \mathcal{Q}_{\mathsf{KGen}}$ *and* $((x_1^0,\ldots,x_n^0),(x_1^1,\ldots,x_n^1))$ *be the predicate submitted to the oracle* KGen *before the challenge phase and the challenge inputs chosen by the adversary. If* $P_i^*(x_i^0)=P_i^*(x_i^1)=0$*, the value* $c_i^{(1)}$ *challenge ciphertext* $c_i=(\widetilde{\mathbb{C}}_i,c_i^{(2)})$ *is computed as* $c_i^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_1,\ldots,x_n),0^{s(\lambda)+k(\lambda)})$ *where* $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$ *(for some function k)* $x_i = x_i^0$*, and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{i\}$*. Otherwise, if* $P_i^*(x_i^0)=P_i^*(x_i^1)=1$*, the value* $c_i^{(1)}$ *challenge ciphertext* $c_i=(\widetilde{\mathbb{C}}_i,c_i^{(2)})$ *is computed as* $c_i^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_1,\ldots,x_n),(y_i,\mathsf{k}_{i+1}))$ *where* $y_i \leftarrow\!\!\$ \leftarrow\!\!\$^{s(\lambda)}$*,* $x_i = x_i^0$*, and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{i\}$*. Observe that* $c_i^{(1)}$ *is computed by fixing* $x_i = x_i^0$ *(instead of* $x_i = x_i^b$*), i.e., the input* $(x_1,\ldots,x_n)$ *used to compute the* i*th challenge ciphertext is fixed and does not depend on the challenge bit b.*

$\mathbf{H}_1^{b,0}(\lambda)$*: Identical to* $\mathbf{H}_1^{b,n}(\lambda)$*.*

$\mathbf{H}_1^{b,i}(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}_1^{b,i-1}$*, except that the challenger changes how it computes the challenger ciphertext* c_i*. Let* $P^* \in \mathcal{Q}_{\mathsf{KGen}}$ *and* $((x_1^0,\ldots,x_n^0),(x_1^1,\ldots,x_n^1))$ *be the predicate submitted to the oracle* KGen *before the challenge phase and the challenge inputs chosen by the adversary. If* $P_i^*(x_i^0)=P_i^*(x_i^1)=0$*, the value* $\widetilde{\mathbb{C}}_i$ *of challenge ciphertext* $c_i=(\widetilde{\mathbb{C}}_i,c_i^{(2)})$ *is simulated by the challenger using the simulator of the lockable obfuscation scheme* LOBF*, i.e.,* $\widetilde{\mathbb{C}}_i \leftarrow\!\!\$\ \mathsf{S}(1^\lambda,1^{|\mathbb{V}_i|},1^{|m_i^b|})$*. Otherwise, if* $P_i^*(x_i^0)=P_i^*(x_i^1)=1$*, the value* $\widetilde{\mathbb{C}}_i$ *is computed as in* $\mathbf{H}_1^{b,0}(\lambda)$*.*

Claim 24. $\mathbf{H}_0^{b,i-1}(\lambda) \approx_c \mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [n]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_0^{b,1-i}(\lambda)$ *and* $\mathbf{H}_0^{b,i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA-2-sided security without collusions of* PE. A *is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ *for* $j \in [n]$. *Let* $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$ *for* $i \in [n]$ *where* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *forward the query* P^* *to* KGen_1 *and return the answer* dk_P.
   - *On input* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, *return* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)}) \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
4. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
5. *Let* $P^*(x_1, \ldots, x_n) = P_1^*(x_1) \wedge \cdots \wedge P_n^*(x_n)$ *be the predicate submitted by* A *to the oracle* KGen. *For any* $j \in [n]$, A *proceeds as follows:*

   Case $j < i$ *and* $P^*(x_j^0) = P^*(x_j^1) = 0$*: Compute* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$.

   Case $j < i$ *and* $P^*(x_j^0) = P^*(x_j^1) = 1$*: Sample* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and execute* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$.

   Case $j = i$ *and* $P^*(x_j^0) = P^*(x_j^1) = 0$*: Send the challenge* $(m_*^0 = (y_i, \mathsf{k}_{i+1}), m_*^1 = 0^{s(\lambda)+k(\lambda)}, x_*^0 = (x_{*1}^0, \ldots, x_{*n}^0), x_*^1 = (x_{*1}^1, \ldots, x_{*n}^1))$ *where* $y_i \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$, $x_{*i}^0 = x_i^b$, $x_{*i}^1 = x_i^0$, *and* $x_{*j}^0 = x_{*j}^1 = x_j^\star$ *for* $j \in [n]\backslash\{i\}$. *Receive the challenge ciphertext* c^* *from the challenger. Set* $c_i^{(1)} = c^*$.

   Case $j = i$ *and* $P^*(x_j^0) = P^*(x_j^1) = 1$*: Send the challenge* $(m_*^0 = (y_i, \mathsf{k}_{i+1}), m_*^1 = (y_i, \mathsf{k}_{i+1}), x_*^0 = (x_{*1}^0, \ldots, x_{*n}^0), x_*^1 = (x_{*1}^1, \ldots, x_{*n}^1))$ *where* $y_i \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$, $x_{*i}^0 = x_i^b$, $x_{*i}^1 = x_i^0$, *and* $x_{*j}^0 = x_{*j}^1 = x_j^\star$ *for* $j \in [n]\backslash\{i\}$. *Receive the challenge ciphertext* c^* *from the challenger. Set* $c_i^{(1)} = c^*$.

   Case $j > i$*: Sample* $y_j \leftarrow\!\!\$\ \leftarrow\!\!\$^{s(\lambda)}$ *and compute* $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j, \mathsf{k}_{j+1}))$ *where* $x_j = x_j^b$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$.

6. *Compute* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *where* $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{ek}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *for any* $j \in [n]$.
7. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
8. *Answer to the incoming oracle queries for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *as in Item* 3.
9. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_0^{b,i-1}(\lambda)$. *On the other hand, if*

$d = 1$, A simulates $\mathbf{H}_0^{b,i}(\lambda)$. Moreover, since D satisfies $\mathbf{Validity}_2$ *and it asks for a single decryption key dk_{P^*} for P^*, we have that either $P_i^*(x_i^0) = P_i^*(x_i^1) = 0$ or $P_i^*(x_i^0) = P_i^*(x_i^1) \wedge m_i^0 = m_i^1$. This implies that A is a valid adversary for the experiment $\mathbf{G}_{\mathsf{PE},\mathsf{A}}^{\mathsf{CPA\text{-}2\text{-}PE}}(\lambda)$ with the same advantage of D. This concludes the proof.* □

Claim 25. $\mathbf{H}_1^{b,i-1}(\lambda) \approx_c \mathbf{H}_1^{b,i}(\lambda)$ *for* $i \in [n]$.

Proof. *Suppose there exists a PPT distinguisher D that distinguishes between $\mathbf{H}_1^{b,1-i}(\lambda)$ and $\mathbf{H}_1^{b,i}(\lambda)$ with non-negligible probability. We build an adversary A that breaks the security of the lockable obfuscation scheme LOBF. A is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$ *for* $j \in [n]$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, \mathsf{k}_{j+1})$. *Let* $\mathsf{k}_{n+1} = \mathsf{k}_1$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_P \leftarrow\$ \mathsf{KGen}(\mathsf{msk}, P)$.
   - *On input* $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, *return* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)}) \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *Let* $P^*(x_1, \ldots, x_n) = P_1^*(x_1) \wedge \ldots \wedge P_n^*(x)$ *be the predicate submitted by* A *to the oracle* KGen. *For any* $j \in [n]$, A *proceeds as follows:*

   Case $j < i$ *and* $P^*(x_j^0) = P^*(x_j^1) = 0$*: Compute* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$. *Finally, set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *where* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{S}(1^\lambda, 1^{|\mathbb{V}_j|}, 1^{|m_j^b|})$.

   Case $j = i$ *and* $P^*(x_j^0) = P^*(x_j^1) = 0$*: Compute* $c_i^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *and* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_i, c_i^{(1)})$ *where* $x_i = x_i^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{i\}$. *Send the challenge* $(\mathbb{V}_i, m_i^b)$ *to the challenger and receive* $\widetilde{\mathbb{C}}_i$. *Set* $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$.

   Case $j > i$ *and* $P^*(x_j^0) = P^*(x_j^1) = 0$*: Compute* $c_j^{(1)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s(\lambda)+k(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$. *Finally, set* $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ *where* $c_j^{(2)} \leftarrow\$ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$ *and* $\widetilde{\mathbb{C}}_j \leftarrow\$ \mathsf{Obf}(1^\lambda, \mathbb{V}_j, y_j, m_j^b)$ *where* $y_j \leftarrow\$ \leftarrow\$^{s(\lambda)}$.

   Case $P^*(x_j^0) = P^*(x_j^1) = 1$*: Compute* $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x_j^0, m_j^b)$.
5. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
6. *Answer to the incoming oracle queries for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *as in Item* 2.
7. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A perfectly simulates the view of D. In particular, if $d = 0$, A simulates $\mathbf{H}_1^{b,i-1}(\lambda)$. On the other hand, if

$d = 1$, A *simulates* $\mathbf{H}_1^{b,i}(\lambda)$. *Hence,* A *retains the same advantage of* D. *This concludes the proof.* □

Claim 26. $\mathbf{H}_1^{b,n}(\lambda) \equiv \mathbf{H}_1^{1-b,n}(\lambda)$.

Proof. *The claim follows by leveraging the validity condition* $\mathbf{Validity}_2$. *Indeed, for every* $i \in [n]$, *if* $P_i^*(x_i^0) = P_i^*(x_i^1) = 0$ *we have that the* j*th ciphertext* c_j *does not depend on the bit* b. *On the other hand, if* $P_i^*(x_i^0) = P_i^*(x_i^1) = 1$, *we have that the* j*th ciphertext* c_j *depends on either* m_j^0 *or* m_j^1. *However, since the adversary satisfies the validity condition* $\mathbf{Validity}_2$ *we have that* $m_j^0 = m_j^1$. *Hence,* $\mathbf{H}_1^{b,n}(\lambda)$ *and* $\mathbf{H}_1^{1-b,n}(\lambda)$ *are identically distributed. This concludes the proof.* □

By combining Claims 24–25 and conditioned to the event $\mathbf{Validity}_2$, we conclude that $\mathbf{H}_0^{0,0} \approx_c \ldots \approx_c \mathbf{H}_0^{0,n} \equiv \mathbf{H}_1^{0,0} \approx_c \ldots \approx_c \mathbf{H}_1^{0,n} \equiv \mathbf{H}_1^{1,n}$. Note that this holds if $n = \mathsf{poly}(\lambda)$ and the adversary is restricted to submitting the (single) key generation query before the challenge phase, i.e., KGen oracle not available after challenge phase. By using complexity leveraging, we conclude that the same result holds also when the KGen oracle is available after the challenge phase when $n = O(\log(\lambda))$. This concludes the proof. □

By leveraging Lemmas 8 and 9, we conclude that Π of Construction 2 is CPA-2-sided secure for $n = O(\log(\lambda))$.

5.3. *Corruption Setting: Multi-input PE from PE, Lockable Obfuscation and PKE*

We now move on to 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 4. (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* $\mathcal{M}_1 = \leftarrow\!\!\$\, ^{m_3(\lambda)+m_4(\lambda)}$, *input space* $\mathcal{X}_1 = \mathcal{X}_{1,1} \times \ldots \times \mathcal{X}_{1,n}$, *and predicate space* $\mathcal{P}_1 = \{P(x_1, \ldots, x_n)\} = \{P_1(x_1) \wedge \ldots \wedge P_n(x_n)\}$. *Without loss of generality, we assume that* 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* $\forall (P_1(x_1) \wedge \ldots \wedge P_n(x_n)) \in \mathcal{P}_1, \forall i \in [n], P_i(x_i^\star) = 1$.
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* 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* $i \in [n] \backslash \{1\}$.
3. *A lockable obfuscation scheme* $\mathsf{LOBF}_3 = (\mathsf{Obf}_3, \mathsf{Eval}_3)$ *with message space* $\mathcal{M}_3 = (\mathcal{K}_{2,1} \cup \ldots \cup \mathcal{K}_{2,n}) \times \leftarrow\!\!\$\, ^{\lfloor \log_2(n) \rfloor + 1}$ *for the family of circuits* $\mathcal{C}^{\mathsf{in}}_{n_3,s_3,d_3}(\lambda)$

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

Initialize:

$c_n = c, \mathsf{sk}'_i = \mathsf{sk}, \mathbb{C}_{n-1} = \bot, k = \bot, \forall j \in [n] \setminus \{i\},\ \mathsf{sk}'_j = \mathsf{sk}_j$

If $\exists w \in [n-2]$ **such that** $\mathbb{C}_w \neq \bot$ **and** $\mathbb{C}_{w+1} = \bot$**:** $k = w$

end initialize.

If $k \neq \bot$ **do:** // If $\mathsf{k} = \bot$, no circuit to execute.

// Execute each circuit received in input in order to retrieve the related secret key.

For $t \in [k]$ **do:**

$\mathsf{Eval}_3(\mathbb{C}_t, (\mathbb{C}_{t+1},\ldots,\mathbb{C}_k, \overbrace{\bot,\ldots,\bot}^{n-2+t-k}, \mathsf{sk}'_1,\ldots,\mathsf{sk}'_n, \mathsf{dk}_P)) = r$

If $r = \bot$**: return** $\bot$

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:** $\mathsf{Dec}_{2,j}(\mathsf{sk}'_j, c_j) = c_{j-1}$

$\mathsf{Dec}_1(\mathsf{dk}_P, c_0) = v$

If $v = \bot$**: return** $\bot$

Else: return y^{in}_i **where** $v = (y^{\mathsf{in}}_i, y^{\mathsf{out}}_i)$

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

Initialize: $c_n = c, \mathsf{sk}'_i = \mathsf{sk}, \forall j \in [n] \setminus \{i\},\ \mathsf{sk}'_j = \bot$

// Execute each circuit received in input in order to retrieve the related secret key.

For t **from** 1 **to** $n-1$ **do:**

$\mathsf{Eval}_3(\mathbb{C}_t, (\mathbb{C}_{t+1},\ldots,\mathbb{C}_{n-1}, \overbrace{\bot,\ldots,\bot}^{t-1}, \mathsf{sk}'_1,\ldots,\mathsf{sk}'_n, \mathsf{dk}_P)) = r$

If $r = \bot$**: return** $\bot$

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:** $\mathsf{Dec}_{2,j}(\mathsf{sk}'_j, c_j) = c_{j-1}$

$\mathsf{Dec}_1(\mathsf{dk}_P, c_0) = v$

If $v = \bot$**: return** $\bot$

Else: return y^{out}_i **where** $v = (y^{\mathsf{in}}_i, y^{\mathsf{out}}_i)$

Fig. 8. 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 LOBF_3 and LOBF_4 of Construction 4.

$= \{\mathbb{C}^{\mathsf{in}}_{c,\mathsf{sk},i}\}$ *defined in Fig. 8, 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. 8, 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 $\mathcal{M} = \overbrace{\mathcal{M}_4 \times \cdots \times \mathcal{M}_4}^{n}$, *input space* $\mathcal{X} = \mathcal{X}_1$, *and predicate space* $\mathcal{P} = \mathcal{P}_1 = \{P(x_1, \ldots, x_n)\} = \{P_1(x_1) \wedge \cdots \wedge P_n(x_n)\}$ *with wildcard (i.e., there exists a (single) wildcard* $(x_1^\star, \ldots, x_n^\star) \in \mathcal{X}$ *such that* $\forall (P_1(x_1) \wedge \ldots \wedge P_n(x_n)) \in \mathcal{P}$, $\forall i \in [n]$, $P_i(x_i^\star) = 1$*), as follows:*

$\mathsf{Setup}(1^\lambda)$*: Upon input the security parameter* 1^λ *the randomized setup algorithm outputs* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ *and* 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* $i \in [n]$.

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

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$*: 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* $(y_i^{\mathsf{in}}, y_i^{\mathsf{out}}) \leftarrow\!\!\$\ \leftarrow\!\!\$^{\,s_3(\lambda)+s_4(\lambda)}$ *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* $j \in [n] \backslash \{i\}$.
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* $\widetilde{\mathbb{C}}_i^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{(n)}_{c_i^{(n)}, \mathsf{sk}_i, i}, y_i^{\mathsf{out}}, m_i)$ *and* $\widetilde{\mathbb{C}}_i^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_i^{(n)}, \mathsf{sk}_i, i}, y_i^{\mathsf{in}}, (\mathsf{sk}_i, i))$.

$\mathsf{Dec}(\mathsf{dk}_P, c_1, \ldots, c_n)$*: Upon input a decryption key* dk_P *for predicate* $P \in \mathcal{P}$*, 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* $m_i = \mathsf{Eval}_4(\widetilde{\mathbb{C}}_i^{\mathsf{out}}, (\widetilde{\mathbb{C}}_1^{\mathsf{in}}, \ldots, \widetilde{\mathbb{C}}_{i-1}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{i+1}^{\mathsf{in}}, \ldots, \widetilde{\mathbb{C}}_n^{\mathsf{in}}, \mathsf{dk}_P))$ *for* $i \in [n]$.

Correctness follows from the one of the underlying primitives (see also Fig. 8 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.

Theorem 7. *Let* $n = O(1)$, PE, $\mathsf{PKE}_{2,1}, \ldots, \mathsf{PKE}_{2,n}$, LOBF_3, *and* LOBF_4 *be as above.*

1. *If* PE *is CPA secure without collusions (Definition 8), each* $\mathsf{PKE}_{2,i}$ *(for* $i \in [n]$*) is CPA secure (Definition 6), and both* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then the n-input PE scheme* Π *from Construction 4 is CPA-1-sided secure in the* $(n-1)$*-corruptions setting without collusions (Definition 13).*
2. *If* PE *is CPA-2-sided secure without collusions (Definition 9), each* $\mathsf{PKE}_{2,i}$ *(for* $i \in [n]$*) is CPA secure (Definition 6), and both* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then the n-input PE scheme* Π *from Construction 4 is CPA-2-sided secure in the* $(n-1)$*-corruptions setting without collusions (Definition 13).*

5.3.1. *Proof of Theorem 7*

CPA-1-sided security of Π (Theorem 7) Consider the predicate space $\mathcal{P} = \{P(x_1, \ldots, x_n)\}$ of Construction 4 where $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \cdots \wedge P_n(x_n)$. Let $P^* \in \mathcal{P}$ be the only predicate for which the adversary will ask the decryption key

dk_{P^*} during the experiment $\mathbf{G}^{(n-1)\text{-CPA-1-iPE}}_{\Pi,\mathsf{A}}$ (recall that we prove the security of Construction 4 in the ℓ-corruptions setting without collusions (i.e., $|\mathcal{Q}_{\mathsf{KGen}}| = 1$). Consider the validity condition of $\mathbf{G}^{(n-1)\text{-CPA-1-iPE}}_{\Pi,\mathsf{A}}$ and let $\mathcal{Q}_i = \{x | \exists (x, m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_i,\cdot,\cdot)}\}$ for $i \in [n] \backslash \mathcal{Q}_{\mathsf{Corr}}$, and $\mathcal{Q}_i = \mathcal{X}_{1,i}$ for $i \in \mathcal{Q}_{\mathsf{Corr}}$ (recall that $|\mathcal{Q}_{\mathsf{Corr}}| \leq n-1$) as defined in Definition 13. We can write such a validity condition with respect to $P^* \in \mathcal{Q}_{\mathsf{KGen}} = \{P^*\}$ as follows: $\forall j \in [n], \forall i_1 \in [k_1+1], \ldots, \forall i_n \in [k_n+1]$,

$$
\begin{aligned}
&P^*(x_1^{(i_1,0)}, \ldots, x_{j-1}^{(i_{j-1},0)}, x_j^0, x_{j+1}^{(i_{j+1},0)}, \ldots, x_n^{(i_n,0)}) \\
&= P^*(x_1^{(i_1,1)}, \ldots, x_{j-1}^{(i_{j-1},1)}, x_j^1, x_{j+1}^{(i_{j+1},1)}, \ldots, x_n^{(i_n,1)}) \\
&= P_1^*(x_1^{(i_1,0)}) \wedge \cdots \wedge P_{j-1}^*(x_{j-1}^{(i_{j-1},0)}) \wedge P_j^*(x_j^0) \wedge P_{j+1}^*(x_{j+1}^{(i_{j+1},0)}) \wedge \cdots \wedge P_{i_n}^*(x_n^{(i_n,0)}) \\
&= P_1^*(x_1^{(i_1,1)}) \wedge \cdots \wedge P_{j-1}^*(x_{j-1}^{(i_{j-1},1)}) \wedge P_j^*(x_j^1) \wedge P_{j+1}^*(x_{j+1}^{(i_{j+1},1)}) \wedge \cdots \wedge P_n^*(x_n^{(i_n,1)}) = 0,
\end{aligned}
$$

where $\mathcal{Q}_i^b = \{x_i^{(1,b)}, \ldots, x_i^{(k_i,b)}, x_i^{(k_i+1,b)} = x_i^b\}$ is the ordered list composed of the k_i predicate inputs $\mathcal{Q}_i$ and the challenge input x_i^b (as defined in Definition 13). Note that Construction 4 has input space $\mathcal{X}_1 = \mathcal{X}_{1,1} \times \cdots \times \mathcal{X}_{1,n}$ (that is identical to the one of the underlying PE). Hence, we can conclude that for each $\mathcal{X}_{1,i}$ for $i \in [n]$ there exists $x_i^\star \in \mathcal{X}_{1,i}$ such that $P_i^*(x_i^\star) = 1$. As a consequence, an adversary is valid only if there exists $j_0, j_1 \in [n] \backslash \mathcal{Q}_{\mathsf{Corr}}$ such that $P_{j_0}^*(x_{j_0}^0) = P_{j_1}^*(x_{j_1}^1) = 0$. Otherwise, an adversary is able to decrypt at least one out the two challenges by leveraging the corrupted encryption keys $\{\mathsf{ek}_i\}_{i \in \mathcal{Q}_{\mathsf{Corr}}}$ and computing $|\mathcal{Q}_{\mathsf{Corr}}|$ ciphertexts, each under the ith predicate wildcard $x_i^\star \in \mathcal{X}_{1,i}$ for $i \in \mathcal{Q}_{\mathsf{Corr}}$.

According to the above observation, the A's validity can be rewritten as follows: $\exists j_0, j_1 \in [n] \backslash \mathcal{Q}_{\mathsf{Corr}}, \forall (x'_1, \ldots, x'_n) \in \mathcal{Q}_1 \times \cdots \times \mathcal{Q}_n$,

$$
\begin{aligned}
&\Big(\Big(P_1^*(x_1^0) = 0 \wedge \cdots \wedge P_n^*(x_n^0) = 0\Big) \vee \Big(P_{j_0}^*(x_{j_0}^0) = 0 \wedge P_{j_0}^*(x'_{j_0}) = 0\Big)\Big) \wedge \\
&\Big(\Big(P_1^*(x_)^1 = 0 \wedge \cdots \wedge P_n^*(x_n^1) = 0\Big) \vee \Big(P_{j_1}^*(x_{j_1}^1) = 0 \wedge P_{j_1}^*(x'_{j_1}) = 0\Big)\Big). \qquad (8)
\end{aligned}
$$

Note that in the above equation we made explicit the challenge inputs and the inputs of $\mathcal{Q}_i$. For this reason, it is enough to quantify over all $(x'_1, \ldots, x'_n) \in \mathcal{Q}_1 \times \cdots \times \mathcal{Q}_n$ where $\mathcal{Q}_i$ is equal to the inputs $\{x_i^{(1)}, \ldots, x_i^{(k_i)}\}$ submitted to oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$, if $i \notin \mathcal{Q}_{\mathsf{Corr}}$. Otherwise (if $i \in \mathcal{Q}_{\mathsf{Corr}}$), $\mathcal{Q}_i$ is equal to the ith input space $\mathcal{X}_{1,i}$. Hence, in order to be valid, A needs to satisfy the condition defined by Eq. (8). This is equivalent to considering the events below: For some $j_0, j_1 \in [n] \setminus \mathcal{Q}_{\mathsf{Corr}}$,[25]

Validity$_1$:

$$P_1^*(x_1^0) = 0 \wedge \cdots \wedge P_n^*(x_n^0) = 0 \wedge P_1^*(x_1^1) = 0 \wedge \cdots \wedge P_n^*(x_n^1) = 0.$$

Validity$_{2,j_0,j_1}$: $\forall x'_{j_0} \in \mathcal{Q}_{j_0}, \forall x'_{j_1} \in \mathcal{Q}_{j_1}$,

[25] Since we are in the $(n-1)$-corruptions setting (i.e, $|\mathcal{Q}_{\mathsf{Corr}}| \leq n-1$) such as $j_0, j_1 \in [n] \setminus \mathcal{Q}_{\mathsf{Corr}}$ always exist.

$$P^*_{j_0}(x^0_{j_0}) = 0 \wedge P^*_{j_0}(x'_{j_0}) = 0 \wedge P^*_{j_1}(x^1_{j_1}) = 0 \wedge P^*_{j_1}(x'_{j_1}) = 0.$$

Validity$_{3,j_0}$: $\forall x'_{j_0} \in \mathcal{Q}_{j_0}$,

$$P^*_{j_0}(x^0_{j_0}) = 0 \wedge P^*_{j_0}(x'_{j_0}) = 0 \wedge P^*_1(x^1_1) = 0 \wedge \cdots \wedge P^*_n(x^1_n) = 0.$$

Validity$_{4,j_1}$: $\forall x'_{j_1} \in \mathcal{Q}_{j_1}$,

$$P^*_1(x^0_1) = 0 \wedge \cdots \wedge P^*_n(x^0_n) = 0 \wedge P^*_{j_1}(x^1_{j_1}) = 0 \wedge P^*_{j_1}(x'_{j_1}) = 0.$$

For the sake of clarity, in the rest of this proof, we use the notation $\mathbb{V}^{\mathsf{in}}_i \stackrel{\text{def}}{=} \mathbb{C}^{\mathsf{in}}_{c_i^{(n)},\mathsf{sk}_i,i}$ (resp. $\mathbb{V}^{\mathsf{out}}_i \stackrel{\text{def}}{=} \mathbb{C}^{\mathsf{out}}_{c_i^{(n)},\mathsf{sk}_i,i}$) where $c_i^{(n)}$, sk_i, and i will be clear from the context. Also, $[a : b]^+_n = \{a, a+1, \ldots, n, 1, 2, \ldots, b\}$. If $1 \le a \le b \le n$, we have $[a : b]^+_n = \{a, a+1, \ldots, b\}$.

Lemma 10. *If* PE *is CPA secure without collusions (Definition 8),* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}^{(n-1)\text{-CPA-1-iPE}}_{\Pi,\mathsf{A}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_1\right] - \frac{1}{2}\right| \le \mathit{negl}(\lambda).$$

Proof. *Consider the following hybrid experiments:*

- $\mathbf{H}^{b,0}_0(\lambda)$*: This is exactly the experiment* $\mathbf{G}^{(n-1)\text{-CPA-1-iPE}}_{\Pi,\mathsf{A}}(\lambda)$ *conditioned to the validity event* $\mathbf{Validity}_1$ *where the challenge bit is b, i.e., the adversary is valid and satisfies* $\mathbf{Validity}_1$.
- $\mathbf{H}^{b,i}_0(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}^{b,i-1}_0$*, except that the challenger changes how it computes the challenger ciphertext* c_i*. Formally, it computes value* $c_i^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *(instead of* $c_i^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y^{\mathsf{in}}_i, y^{\mathsf{out}}_i))$*) where* $c_i^{(0)}$ *is the value used to compute the challenge ciphertext* $x_i = x_i^0$*, and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{i\}$*. Observe that* c_i *is computed by fixing* $x_i = x_i^0$ *(instead of* $x_i = x_i^b$*), i.e., the predicate input* $(x_1, \ldots, x_n)$ *used to compute the* i*th challenge ciphertext is fixed and does not depend on the challenge bit b.*
- $\mathbf{H}^{b,0}_1(\lambda)$*: Identical to* $\mathbf{H}^{b,n}_0(\lambda)$.
- $\mathbf{H}^{b,i}_1(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}^{b,i-1}_1$*, except that the challenger changes how it computes the challenger ciphertext* c_i*. Formally, the value* $\widetilde{\mathbb{C}}^{\mathsf{in}}_i$ *of challenge ciphertext* $c_i = (\widetilde{\mathbb{C}}^{\mathsf{in}}_i, \widetilde{\mathbb{C}}^{\mathsf{out}}_i)$ *is simulated by the challenger using the simulator* S_3 *of the lockable obfuscation scheme* LOBF_3*, i.e.,* $\mathsf{S}(1^\lambda, 1^{|\mathbb{V}^{\mathsf{in}}_i|}, 1^{|(\mathsf{sk}_i,i)|})$.
- $\mathbf{H}^{b,0}_2(\lambda)$*: Identical to* $\mathbf{H}^{b,n}_1(\lambda)$.
- $\mathbf{H}^{b,i}_2(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}^{b,i-1}_2$*, except that the challenger changes how it computes the challenger ciphertext* c_i*. Formally, the value* $\widetilde{\mathbb{C}}^{\mathsf{out}}_i$ *of challenge ciphertext* $c_i = (\widetilde{\mathbb{C}}^{\mathsf{in}}_i, \widetilde{\mathbb{C}}^{\mathsf{out}}_i)$ *is simulated by the challenger using the simulator* S_4 *of the lockable obfuscation scheme* LOBF_4*, i.e.,* $\mathsf{S}(1^\lambda, 1^{|\mathbb{V}^{\mathsf{out}}_i|}, 1^{|m^b_i|})$.

Claim 27. $\mathbf{H}_0^{b,i-1}(\lambda) \approx_c \mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [n]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_0^{b,1-i}(\lambda)$ *and* $\mathbf{H}_0^{b,i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA security without collusions of* PE. A *is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow\!\!\$\ \mathsf{KGen}_{2,j}(1^\lambda)$ *and set* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ *for* $j \in [n]$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen*, forward the query* P^* *to* KGen_1 *and return the answer* dk_{P^*}.
   - *On input* $j \in [n]$ *for* Corr*, return* ek_j.
   - *On input* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$*, return* $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
4. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
5. *For any* $j \in [n]$*,* A *proceeds as follows:*

   Case $j < i$*: Sample* $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \leftarrow\!\!\$^{s_3(\lambda)+s_4(\lambda)}$. *Compute* $c_j^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_j = x_j^0$*, and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$.

   Case $j = i$*: Send the challenge* $(m^0 = (y_i^{\mathsf{in}}, y_i^{\mathsf{out}}), m^1 = 0^{s_3(\lambda)+s_4(\lambda)}, x = (x_1, \ldots, x_n))$ *where* $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \leftarrow\!\!\$^{s_3(\lambda)+s_4(\lambda)}$*,* $x_i = x_i^b$*, and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{i\}$*. Receive the challenge ciphertext* c^* *from the challenger. Set* $c_i^{(0)} = c^*$.

   Case $j > i$*: Sample* $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \leftarrow\!\!\$^{s_3(\lambda)+s_4(\lambda)}$. *Compute* $c_j^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j^{\mathsf{in}}, y_j^{\mathsf{out}}))$ *where* $x_j = x_j^0$*, and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$.
6. *For every* $j \in [n]$*, compute* $c_j^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ *for* $v \in [n]$.
7. *Compute* $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_j^{\mathsf{in}}, y_j^{\mathsf{in}}, (\mathsf{sk}_j, j))$ *and* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_j^{\mathsf{out}}, y_j^{\mathsf{out}}, m_j^b)$ *for any* $j \in [n]$.
8. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
9. *Answer to the incoming oracle queries as in Item* 3.
10. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D*. In particular, if* $d = 0$*,* A *simulates* $\mathbf{H}_0^{b,i-1}(\lambda)$*. On the other hand, if* $d = 1$*,* A *simulates* $\mathbf{H}_1^{b,i}(\lambda)$*. Moreover, since* D *satisfies* $\mathbf{Validity}_1$ *and it asks for a single decryption key* dk_{P^*} *for* P^**, we have that* $P_i^*(x_i^0) = 0 \wedge P_i^*(x_i^1) = 0$*. Because of this,* A *submits a single query* P^* *to oracle* KGen_1 *and it is also a valid adversary for the experiment* $\mathbf{G}_{\mathsf{PE},\mathsf{A}}^{\mathsf{CPA\text{-}PE}}(\lambda)$ *with the same advantage of* D*. This concludes the proof.* □

Claim 28. $\mathbf{H}_1^{b,i-1}(\lambda) \approx_c \mathbf{H}_1^{b,i}(\lambda)$ *for* $i \in [n]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_1^{b,1-i}(\lambda)$ *and* $\mathbf{H}_1^{b,i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF_3. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ *for* $j \in [n]$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, P^*)$.
   - *On input* $j \in [n]$ *for* Corr, *return* ek_j.
   - *On input* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$, *return* $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$, *run* $c_j^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\backslash\{j\}$.
5. *For every* $j \in [n]$, *compute* $c_j^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ *for* $v \in [n]$.
6. *For any* $j \in [n]$, A *proceeds as follows:*

   Case $j < i$: *Compute* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_j, j)|})$.

   Case $j = i$: *Send the challenge* $(\mathbb{V}_i^{\mathsf{in}}, (\mathsf{sk}_i, i))$ *to the challenger and receive* $\widetilde{\mathbb{C}}_i^{\mathsf{in}}$.

   Case $j > i$: *Compute* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_j^{\mathsf{in}}, y_j^{\mathsf{in}}, (\mathsf{sk}_j, j))$ *where* $y_j^{\mathsf{in}} \leftarrow\!\!\$\ \leftarrow\!\!\$^{s_3(\lambda)}$.
7. *For every* $j \in [n]$, *compute* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_j^{\mathsf{out}}, y_j^{\mathsf{out}}, m_j^b)$ *where* $y_j^{\mathsf{out}} \leftarrow\!\!\$\ \leftarrow\!\!\$^{s_4(\lambda)}$.
8. *Set* $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
9. *Answer to the incoming oracle queries as in Item* 2.
10. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_1^{b,i-1}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_1^{b,i}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 29. $\mathbf{H}_2^{b,i-1}(\lambda) \approx_c \mathbf{H}_2^{b,i}(\lambda)$ *for* $i \in [n]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_2^{b,1-i}(\lambda)$ *and* $\mathbf{H}_2^{b,i}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF_4. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ *for* $j \in [n]$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, P^*)$.
   - *On input* $j \in [n]$ *for* Corr, *return* ek_j.

- *On input* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$, *return* $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.

3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$, *compute* $c_j^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_j = x_j^0$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n] \backslash \{j\}$.
5. *For every* $j \in [n]$, *run* $c_j^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ *for* $v \in [n]$.
6. *For every* $j \in [n]$, *compute* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_j, j)|})$.
7. *For every* $j \in [n]$, A *proceeds as follows:*

   Case $j < i$: *Compute* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{out}}|}, 1^{|m_j^b|})$.
   Case $j = i$: *Send the challenge* $(\mathbb{V}_i^{\mathsf{out}}, m_i^b)$ *to the challenger and receive* $\widetilde{\mathbb{C}}_i^{\mathsf{out}}$.
   Case $j > i$: *Compute* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_j^{\mathsf{out}}, y_j^{\mathsf{out}}, m_j^b)$ *where* $y_j^{\mathsf{out}} \leftarrow\!\!\$\ \leftarrow\!\!\$^{\,s_4(\lambda)}$.
8. *Set* $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *for* $j \in [n]$ *and send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
9. *Answer to the incoming oracle queries as in Item* 2.
10. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_2^{b,i-1}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_2^{b,i}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 30. $\mathbf{H}_2^{b,n}(\lambda) \equiv \mathbf{H}_2^{1-b,n}(\lambda)$.

Proof. *The distribution of these two experiments does not depend on the bit* b. □

By combining Claims 27–30 and conditioned to the event $\mathbf{Validity}_1$, we conclude that

$$\mathbf{H}_0^{b,0} \approx_c \ldots \approx_c \mathbf{H}_0^{b,n} \equiv \mathbf{H}_1^{b,0} \ldots \approx_c \mathbf{H}_1^{b,n} \equiv \mathbf{H}_2^{b,0} \approx_c \ldots \approx_c \mathbf{H}_2^{b,n} \equiv \mathbf{H}_2^{1-b,n}.$$

This concludes the proof. □

Lemma 11. *Let* $j_0, j_1 \in [n] \setminus \mathcal{Q}_{\mathsf{Corr}}$. *If* PE *is CPA secure without collusions (Definition 8),* PKE_{2,j_0} *and* PKE_{2,j_1} *are CPA secure (Definition 6),* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then*

$$\left| \mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{(n-1)\text{-}\mathsf{CPA}\text{-}1\text{-}\mathsf{iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_{2,j_0,j_1} \right] - \frac{1}{2} \right| \leq \mathit{negl}(\lambda).$$

Proof. *Without loss of generality, let* $q = |\mathcal{Q}_{j_0}| = |\mathcal{Q}_{j_1}| \in \mathit{poly}(\lambda)$ *(recall* $j_0, j_1 \notin \mathcal{Q}_{\mathsf{Corr}}$*). Consider the following hybrid experiments:*

$\mathbf{H}_0^b(\lambda)$: *This is exactly the experiment* $\mathbf{G}_{\Pi,\mathsf{A}}^{(n-1)\text{-}\mathsf{CPA}\text{-}1\text{-}\mathsf{iPE}}(\lambda)$ *conditioned to the validity event* $\mathbf{Validity}_{2,j_0,j_1}$ *where the challenge bit is* b, *i.e., the adversary is valid and satisfies the validity event* $\mathbf{Validity}_{2,j_0,j_1}$.

$\mathbf{H}_1^b(\lambda)$: *Same as* $\mathbf{H}_0^b$*, except that the challenger changes how it computes the challenge* j_b*th ciphertext* c_{j_b}*. Specifically, it computes* $c_{j_b}^{(0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *(instead of* $c_{j_b}^{(0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_i^{\mathsf{in}}, y_i^{\mathsf{out}}))$*) where the value* $c_{j_b}^{(0)}$ *is used to compute the challenge ciphertext,* $x_i = x_i^b$*, and* $x_j = x_j^\star$ *for* $j \in [n] \backslash \{j_b\}$.

$\mathbf{H}_2^{b,0}$: *Identical to* $\mathbf{H}_1^b(\lambda)$.

$\mathbf{H}_2^{b,i}(\lambda)$ *for* $i \in [q]$: *Same as* $\mathbf{H}_2^{b,i-1}(\lambda)$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_b}, \cdot, \cdot)$*. Formally, on input the* i'*th query* (x, m) *such that* $i' \leq i$*, the challenger computes* $c_{j_b}^{(0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x$*, and* $x_j = x_j^\star$ *for* $j \in [n] \backslash \{j_b\}$*. Finally, the challenger returns* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *where* $c_{j_b}^{(v)} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_b}^{(b-1)})$ *for* $v \in [n]$, $(y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$, $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\$ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{in}}, (\mathsf{sk}_{j_b}, j_b))$*, and* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_{j_b}^{\mathsf{out}}, y_{j_b}^{\mathsf{out}}, m_{j_b}^b)$*. Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_2^{b,0}$.

$\mathbf{H}_3^b(\lambda)$: *Same as* $\mathbf{H}_2^{b,q}$*, except that the challenger changes how it computes the challenge* j_b*th ciphertext* c_{j_b}*. Formally, the value* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}$ *of challenge* j_b*th ciphertext* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *is simulated by the challenger using the simulator* S_3 *of the lockable obfuscation scheme* LOBF_3*, i.e.,* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}$ *is computed by executing* $\mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_{j_b}^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_{j_b}, j_b)|})$.

$\mathbf{H}_4^b(\lambda)$: *Same as* $\mathbf{H}_3^b$*, except that the challenger changes how it computes the challenge* j_b*th ciphertext* c_{j_b}*. Formally, the value* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}}$ *of challenge* j_b*th ciphertext* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *is simulated by the challenger using the simulator* S_4 *of the lockable obfuscation scheme* LOBF_4*, i.e.,* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}}$ *is computed by executing* $\mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_{j_b}^{\mathsf{out}}|}, 1^{|m_{j_b}^b|})$.

$\mathbf{H}_5^{b,0}$: *Identical to* $\mathbf{H}_4^b(\lambda)$.

$\mathbf{H}_5^{b,i}(\lambda)$ *for* $i \in [q]$: *Same as* $\mathbf{H}_5^{b,i-1}(\lambda)$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_b}, \cdot, \cdot)$*. Formally, on input the* i'*th query* (x, m) *such that* $i' \leq i$*, the challenger returns* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}$ *is computed using the simulator* S_3 *of the lockable obfuscator scheme* LOBF_3*, i.e.,* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\$ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_{j_b}^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_{j_b}, j_b)|})$*. Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_5^{b,0}$.

$\mathbf{H}_6^{b,0}$: *Identical to* $\mathbf{H}_5^{b,q}(\lambda)$.

$\mathbf{H}_6^{b,i}(\lambda)$ *for* $i \in [q]$: *Same as* $\mathbf{H}_6^{b,i-1}(\lambda)$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_b}, \cdot, \cdot)$*. Formally, on input the* i'*th query* (x, m) *such that* $i' \leq i$*, the challenger returns* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}}$ *is computed using the simulator* S_4 *of the lockable obfuscator scheme* LOBF_4*, i.e.,* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\$ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_{j_b}^{\mathsf{out}}|}, 1^{|m|})$*. Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_6^{b,0}$.

$\mathbf{H}_6^{b,1,1}$: *Identical to* $\mathbf{H}_6^{b,q}(\lambda)$.

$\mathbf{H}_{7+i}^{b,0,0}$ *for* $i \in \{0\} \cup [n-2]$: *Same as* $\mathbf{H}_{7+i-1}^{b,1,1}$ *except that the challenger changes how it computes the challenge ciphertext* c_r *where* $r = (j_b + i \mod n) + 1$. *Formally, the value* $c_r^{(j_b)}$ *is computed as* $c_r^{(j_b)} \leftarrow\$ \mathsf{Enc}_{2,j_b}(\mathsf{pk}_{j_b}, w)$ *where* $w \leftarrow\$ \mathcal{M}_{2,j_b}$.

$\mathbf{H}_{7+i}^{b,1,0}$ *for* $i \in \{0\} \cup [n-2]$: *Same as* $\mathbf{H}_{7+i}^{b,1,0}(\lambda)$ *except that the challenger changes how it computes the challenge ciphertext* c_r *where* $r = (j_b + i \mod n) + 1$. *Formally, the value* $\widetilde{\mathbb{C}}_v^{\mathsf{in}}$ *of challenge ciphertext* $c_r = (\widetilde{\mathbb{C}}_r^{\mathsf{in}}, \widetilde{\mathbb{C}}_r^{\mathsf{out}})$ *is simulated by the challenger using the simulator of the lockable obfuscation scheme* LOBF_3, *i.e.,* $\widetilde{\mathbb{C}}_v^{\mathsf{in}}$ *is computed by executing* $\mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_r^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_r, r)|})$.

$\mathbf{H}_{7+i}^{b,1,1}$ *for* $i \in \{0\} \cup [n-2]$: *Same as* $\mathbf{H}_{7+i}^{b,1,0}(\lambda)$ *except that the challenger changes how it computes the challenge ciphertext* c_r *where* $r = (j_b + i \mod n) + 1$. *Formally, the value* $\widetilde{\mathbb{C}}_v^{\mathsf{out}}$ *of challenge ciphertext* $c_r = (\widetilde{\mathbb{C}}_r^{\mathsf{in}}, \widetilde{\mathbb{C}}_r^{\mathsf{out}})$ *is simulated by the challenger using the simulator of the lockable obfuscation scheme* LOBF_4, *i.e.,* $\widetilde{\mathbb{C}}_v^{\mathsf{out}}$ *is computed by executing* $\mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_r^{\mathsf{out}}|}, 1^{|m_r^b|})$.

Claim 31. $\mathbf{H}_0^b(\lambda) \approx_c \mathbf{H}_1^b(\lambda)$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_0^b(\lambda)$ *and* $\mathbf{H}_1^b(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA security without collusions of* PE. A *is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow\$ \mathsf{KGen}_{2,j}(1^\lambda)$ *and set* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ *for* $j \in [n]$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *forward the query* P^* *to* KGen_1 *and return the answer* dk_{P^*}.
   - *On input* $j \in [n]$ *for* Corr, *return* ek_j.
   - *On input* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$, *return* $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
4. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D. *Send the challenge* $(m^0 = (y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}), m^1 = 0^{s_3(\lambda)+s_4(\lambda)}, x = (x_1, \ldots, x_n))$ *where* $(y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$, $x_{j_b} = x_{j_b}^b$ *and* $x_j = x_j^\star$ *for* $j \in [n] \backslash \{j_b\}$.
5. *Receive the challenge ciphertext* c^* *from the challenger. Set* $c_{j_b}^{(0)} = c^*$.
6. *For every* $j \in [n] \setminus \{j_b\}$, *compute* $c_j^{(0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j^{\mathsf{in}}, y_j^{\mathsf{out}}))$ *where* $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$, $x_j = x_j^b$, *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n] \backslash \{j\}$.
7. *For every* $j \in [n]$, *compute* $c_j^{(v)} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ *for* $v \in [n]$.
8. *Compute* $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\$ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_j^{\mathsf{in}}, y_j^{\mathsf{in}}, (\mathsf{sk}_j, j))$ *and* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_j^{\mathsf{out}}, y_j^{\mathsf{out}}, m_j^b)$ *for any* $j \in [n]$.
9. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
10. *Answer to the incoming oracle queries as in Item* 3.
11. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A perfectly simulates the view of D. In particular, if $d = 0$, A simulates $\mathbf{H}_0^b(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}_1^b(\lambda)$. Moreover, D submits a single query P^* to oracle KGen and it satisfies the validity condition $\mathbf{Validity}_{2,j_0,j_1}$, we know that $P^*_{j_b}(x^b_{j_b}) = 0$. Because of this, A submits a single query to oracle KGen_1 and, it is also a valid adversary for the experiment $\mathbf{G}^{\mathsf{CPA\text{-}PE}}_{\mathsf{PE},\mathsf{A}}(\lambda)$ with the same advantage of D. This concludes the proof. □

Claim 32. $\mathbf{H}_2^{b,i-1}(\lambda) \approx_c \mathbf{H}_2^{b,i}(\lambda)$ *for* $i \in [q]$.

Proof. *Suppose there exists a PPT distinguisher D that distinguishes between $\mathbf{H}_2^{b,i-1}(\lambda)$ and $\mathbf{H}_2^{b,i}(\lambda)$ with non-negligible probability. We build an adversary A that breaks the CPA security without collusions of PE. A is defined as follows:*

1. *Receive* mpk *from the challenger.*
2. *Compute* $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow\$ \mathsf{KGen}_{2,j}(1^\lambda)$ *and set* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ *for* $j \in [n]$.
3. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *forward the query* P^* *to* KGen_1 *and return the answer* dk_{P^*}.
   - *On input* $j \in [n]$ *for* Corr, *return* ek_j.
   - *On input* i'*th query* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$, A *proceeds as follows:*

     Case $j \neq j_b$*: Sample* $(y^{\mathsf{in}}_j y^{\mathsf{out}}_j) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$. *Run* $c^{(0)}_j \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y^{\mathsf{in}}_j, y^{\mathsf{out}}_j))$ *where* $x_j = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j\}$.

     Case $j = j_b$ *and* $i' < i$*: Sample* $(y^{\mathsf{in}}_{j_b}, y^{\mathsf{out}}_{j_b}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$. *Compute* $c^{(0)}_{j_b} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$.

     Case $j = j_b$ *and* $i' = i$*: Sample* $(y^{\mathsf{in}}_{j_b}, y^{\mathsf{out}}_{j_b}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$. *Send the challenge* $(m^0 = (y^{\mathsf{in}}_{j_b}, y^{\mathsf{out}}_{j_b}), m^1 = 0^{s_3(\lambda)+s_4(\lambda)}, x = (x_1, \ldots, x_n))$ *to the challenger where* $x_{j_b} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$. *Receive the challenge ciphertext* c^* *and set* $c^{(0)}_{j_b} = c^*$.

     Case $j = j_b$ *and* $i' > i$*: Sample* $(y^{\mathsf{in}}_j, y^{\mathsf{out}}_j) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$. *Compute* $c^{(-1)}_{j_b} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y^{\mathsf{in}}_j, y^{\mathsf{out}}_j))$ *where* $x_{j_b} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}^{\mathsf{in}}_j, \widetilde{\mathbb{C}}^{\mathsf{out}}_j)$ *where* $c^{(v)}_j \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c^{(v-1)}_j)$ *for* $v \in [n]$, $\widetilde{\mathbb{C}}^{\mathsf{in}}_j \leftarrow\$ \mathsf{Obf}_3(1^\lambda, \mathbb{V}^{\mathsf{in}}_j, y^{\mathsf{in}}_j, (\mathsf{sk}_j, j))$ *and* $\widetilde{\mathbb{C}}^{\mathsf{out}}_j \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}^{\mathsf{out}}_j, y^{\mathsf{out}}_j, m)$.
4. *Receive the challenge* $((m^0_1, \ldots, m^0_n), (m^1_1, \ldots, m^1_n), (x^0_1, \ldots, x^0_n), (x^1_1, \ldots, x^1_n))$ *from* D.
5. *Compute* $c^{(0)}_{j_b} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x^b_{j_b}$, $x_{j'} = x^\star_{j'}$ *for* $j' \in [n]\backslash\{j_b\}$, *and* $(y^{\mathsf{in}}_{j_b}, y^{\mathsf{out}}_{j_b}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$.

6. *Compute* $c_{j_b}^{(v)} \leftarrow_{\$} \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_b}^{(v-1)})$ *for* $v \in [n]$.
7. *Compute* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow_{\$} \mathsf{Obf}_3(1^\lambda, \mathbb{V}_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{in}}, (\mathsf{sk}_{j_b}, j_b))$ *and* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow_{\$} \mathsf{Obf}_4(1^\lambda, \mathbb{V}_{j_b}^{\mathsf{out}}, y_{j_b}^{\mathsf{out}}, m_{j_b}^b)$.
8. *For every* $j \in [n] \setminus \{j_b\}$, *compute* $c_j \leftarrow_{\$} \mathsf{Enc}(\mathsf{ek}_j, x_j^b, m_j^b)$.
9. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
10. *Answer to the incoming oracle queries as in Item* 3.
11. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_2^{b,i-1}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_2^{b,i}(\lambda)$. *Moreover, since* D *submits a single query* P^* *to oracle* KGen *and it satisfies the validity condition* $\mathbf{Validity}_{2,j_0,j_1}$, *we have that* $j_b \notin \mathcal{Q}_{\mathsf{Corr}}$ *and* $\forall x'_{j_b} \in \mathcal{Q}_{j_b} \subset \mathcal{X}_{1,j_b}$, $P^*_{j_b}(x'_{j_b}) = 0$. *Because of this,* A *submits a single query to oracle* KGen_1 *and it is also a valid adversary for the experiment* $\mathbf{G}_{\mathsf{PE},\mathsf{A}}^{\mathsf{CPA\text{-}PE}}(\lambda)$ *with the same advantage of* D. *This concludes the proof.* □

Claim 33. $\mathbf{H}_2^{b,q}(\lambda) \approx_c \mathbf{H}_3^b(\lambda)$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_2^{b,q}(\lambda)$ *and* $\mathbf{H}_3^b(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF_3. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ *for* $j \in [n]$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow_{\$} \mathsf{KGen}(\mathsf{msk}, P^*)$.
   - *On input* $j \in [n]$ *for* Corr, *return* ek_j.
   - *On input* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j = j_b$: *Sample* $(y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}) \leftarrow_{\$} \leftarrow_{\$}^{s_3(\lambda)+s_4(\lambda)}$. *Run* $c_{j_b}^{(0)} \leftarrow_{\$} \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\setminus\{j_b\}$.

     Case $j \neq j_b$: *Compute* $c_j^{(0)} \leftarrow_{\$} \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_j^{\mathsf{in}}, y_j^{\mathsf{out}}))$ *where* $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow_{\$} \leftarrow_{\$}^{s_3(\lambda)+s_4(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\setminus\{j\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *where* $c_j^{(v)} \leftarrow_{\$} \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ *for* $v \in [n]$, $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow_{\$} \mathsf{Obf}_3(1^\lambda, \mathbb{V}_j^{\mathsf{in}}, y_j^{\mathsf{in}}, (\mathsf{sk}_j, j))$ *and* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow_{\$} \mathsf{Obf}_4(1^\lambda, \mathbb{V}_j^{\mathsf{out}}, y_j^{\mathsf{out}}, m)$.
3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *Compute* $c_{j_b}^{(0)} \leftarrow_{\$} \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x_{j_b}^b$ *and* $x_{j'} = x_{j'}^\star$ *for* $j' \in [n]\setminus\{j_b\}$.
5. *Compute* $c_{j_b}^{(v)} \leftarrow_{\$} \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_b}^{(v-1)})$ *for* $v \in [n]$.

6. *Send the challenge* $(\mathbb{V}^{\mathsf{in}}_{j_b}, (\mathsf{sk}_{j_b}, j_b))$ *to the challenger and receive* $\widetilde{\mathbb{C}}$. *Compute* $c_{j_b} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b})$ *where* $\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b} = \widetilde{\mathbb{C}}$, $\widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}^{\mathsf{out}}_{j_b}, y^{\mathsf{out}}_{j_b}, m^b_{j_b})$ *and* $y^{\mathsf{out}}_{j_b} \leftarrow\$ \leftarrow\$^{s_4(\lambda)}$.
7. *For every* $j \in [n] \setminus \{j_b\}$, *compute* $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x^b_j, m^b_j)$.
8. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
9. *Answer to the incoming oracle queries as in Item* 2.
10. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}^{b,q}_2(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}^b_3(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 34. $\mathbf{H}^{b,i-1}_4(\lambda) \approx_c \mathbf{H}^{b,i}_4(\lambda)$.

Proof. *Claim 34 follows by leveraging a similar argument to that of Claim 33.* □

Claim 35. $\mathbf{H}^{b,i-1}_5(\lambda) \approx_c \mathbf{H}^{b,i}_5(\lambda)$ *for* $i \in [q]$.

Proof. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}^{b,i-1}_5(\lambda)$ *and* $\mathbf{H}^{b,i}_5(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF_3. A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *forward the query* P^* *to* KGen_1 *and return the answer* dk_{P^*}.
   - *On input* $j \in [n]$ *for* Corr, *return* ek_j.
   - *On input* i'*th query* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ *where* $j \in [n]$, A *proceeds as follows:*

     Case $j \neq j_b$: *Return* $c_j = (\widetilde{\mathbb{C}}^{\mathsf{in}}_j, \widetilde{\mathbb{C}}^{\mathsf{out}}_j) \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.

     Case $j = j_b$ *and* $i' < i$: *Sample* $y^{\mathsf{out}}_{j_b} \leftarrow\$ \leftarrow\$^{s_4(\lambda)}$. *Run* $c^{(0)}_{j_b} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n] \setminus \{j_b\}$. *Return* $c_{j_b} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b})$ *where* $c^{(v)}_{j_b} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c^{(v-1)}_{j_b})$ *for* $v \in [n]$, $\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b} \leftarrow\$ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}^{\mathsf{in}}_{j_b}|}, 1^{|(\mathsf{sk}_{j_b}, j_b)|})$, *and* $\widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}^{\mathsf{out}}_{j_b}, y^{\mathsf{out}}_{j_b}, m)$.

     Case $j = j_b$ *and* $i' = i$: *Sample* $y^{\mathsf{out}}_{j_b} \leftarrow\$ \leftarrow\$^{s_4(\lambda)}$. *Run* $c^{(0)}_{j_b} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x$ *and* $x_{j'} = x^\star_{j'}$ *for* $j' \in [n] \setminus \{j_b\}$. *Send the challenge* $(\mathbb{V}^{\mathsf{in}}_{j_b}, (\mathsf{sk}_{j_b}, j_b))$ *to the challenger where* $c^{(v)}_{j_b} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c^{(v-1)}_{j_b})$ *for* $v \in [n]$. *Receive the challenge ciphertext* $\widetilde{\mathbb{C}}$ *and set* $\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b} = \widetilde{\mathbb{C}}$. *Return* $c_{j_b} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b})$ *where* $\widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}^{\mathsf{out}}_{j_b}, y^{\mathsf{out}}_{j_b}, m)$.

Case $j = j_b$ and $i' > i$: Sample $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$. *Compute* $c_{j_b}^{(0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_{j_b} = x$ *and* $x_{j'} = x_{j'}^{\star}$ *for* $j' \in [n]\backslash\{j_b\}$. *Return* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *where* $c_{j_b}^{(v)} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_b}^{(v-1)})$ *for* $v \in [n]$, $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\$ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{in}}, (\mathsf{sk}_{j_b}, j_b))$, *and* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_{j_b}^{\mathsf{out}}, y_{j_b}^{\mathsf{out}}, m)$.

3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *Compute* $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\$ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_{j_b}^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_{j_b}, j_b)|})$ *and* $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\$ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_{j_b}^{\mathsf{out}}|}, 1^{|m_{j_b}^b|})$.
5. *For every* $j \in [n] \setminus \{j_b\}$, *compute* $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x_j^b, m_j^b)$.
6. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
7. *Answer to the incoming oracle queries as in Item* 2.
8. *Return the output of* D.

Let d be the challenge bit sampled by the challenger. The adversary A *perfectly simulates the view of* D. *In particular, if* $d = 0$, A *simulates* $\mathbf{H}_5^{b,i-1}(\lambda)$. *On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_5^{b,i}(\lambda)$. *Hence,* A *has the same advantage of* D. *This concludes the proof.* □

Claim 36. $\mathbf{H}_6^{b,i-1}(\lambda) \approx_c \mathbf{H}_6^{b,i}(\lambda)$ *for* $i \in [q]$.

Proof. *Claim 36 follows by leveraging a similar argument to that of Claim 35.* □

Claim 37. $\mathbf{H}_{7+i-1}^{b,1,1}(\lambda) \approx_c \mathbf{H}_{7+i}^{b,0,0}(\lambda)$ *for* $i \in \{0\} \cup [n-2]$.

Proof. *Let* $r = (j_b + i \mod n) + 1$. *Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_{7+i-1}^{b,1,1}(\lambda)$ *and* $\mathbf{H}_{7+i}^{b,0,0}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the CPA security of* PKE_{2,j_b}. A *is defined as follows:*

1. *Compute* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}_1(1^\lambda)$ *and* $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow\$ \mathsf{KGen}_{2,j}(1^\lambda)$ *for* $j \in [n]\backslash\{j_b\}$.
2. *Receive* pk_{j_b} *from the challenger.*
3. *Set* $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ *for* $j \in [n]\backslash\{j_b\}$.
4. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen, *return* $\mathsf{dk}_{P^*} \leftarrow\$ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
   - *On input* $j \in [n]$ *for* Corr, *return* ek_j.
   - *On input* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$, A *proceeds as follows:*

     Case $j = j_b$: Run $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\$ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_j, j)|})$ *and* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\$ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{out}}|}, 1^{|m|})$.
     Case $j \neq j_b$: Compute $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.

     Finally, return c_j.
5. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.

6. *For every* $j \in [n]$*, the adversary* A *proceeds as follows:*

   Case $j \in [j_b : r-1]_n^+$*: Compute* $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\$ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_j, j)|})$ *and* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\$ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{out}}|}, 1^{|m_j^b|})$.

   Case $j = r$*: Sample* $(y_r^{\mathsf{in}}, y_r^{\mathsf{out}}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$ *and compute* $c_r^{(0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_r^{\mathsf{in}}, y_r^{\mathsf{out}}))$ *where* $x_r = x_r^b$*,* $x_{j'} = x_{j'}^\star$ *for any* $j' \in [n]\backslash\{r\}$*. Compute* $c_r^{(v)} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_r^{(v-1)})$ *for* $v \in [j_b - 1]$*. Send the challenge* $(m^0 = c_r^{(v)}, m^1 = w)$ *to the challenger where* $w \leftarrow\$ \mathcal{M}_{2,j_b}$*. Receive the answer* c^* *and set* $c_r^{(j_b)} = c^*$*. Compute* $c_r^{(v)} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_r^{(v-1)})$ *for* $v \in [n]\backslash[j_b]$*. Set* $c_r = (\widetilde{\mathbb{C}}_r^{\mathsf{in}}, \widetilde{\mathbb{C}}_r^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_r^{\mathsf{in}} \leftarrow\$ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_r^{\mathsf{in}}, y_r^{\mathsf{in}}, (\mathsf{sk}_r, r))$ *and* $\widetilde{\mathbb{C}}_r^{\mathsf{out}} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_r^{\mathsf{out}}, y_r^{\mathsf{out}}, m_r^b)$.

   Case $i < n-2$ *and* $j \in [r+1 : j_b - 1]_n^+$*: Compute* $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x_j^b, m_j^b)$.

7. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
8. *Answer to the incoming oracle queries as in Item* 4.
9. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D*. This is because, by the* $\mathbf{Validity}_{2,j_0,j_1}$ *we have that* $j_b \notin \mathcal{Q}_{\mathsf{Corr}}$*, i.e.,* A *can simulate the view of* D *without knowing* sk_{j_b} *(sampled by the challenger). Moreover, if* $d = 0$*,* A *simulates* $\mathbf{H}_{7+i-1}^{b,1,1}(\lambda)$*. On the other hand, if* $d = 1$*,* A *simulates* $\mathbf{H}_{7+i}^{b,0,0}(\lambda)$*. Hence,* A *has the same advantage of* D*. This concludes the proof.* □

Claim 38. $\mathbf{H}_{7+i}^{b,0,0}(\lambda) \approx_c \mathbf{H}_{7+i}^{b,1,0}(\lambda)$ *for* $i \in \{0\} \cup [n-2]$.

Proof. *Let* $r = (j_b + i \mod n) + 1$*. Suppose there exists a PPT distinguisher* D *that distinguishes between* $\mathbf{H}_{7+i}^{b,0,0}(\lambda)$ *and* $\mathbf{H}_{7+i}^{b,1,0}(\lambda)$ *with non-negligible probability. We build an adversary* A *that breaks the security of the lockable obfuscation scheme* LOBF_3*.* A *is defined as follows:*

1. *Compute* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$.
2. A *answers to the incoming oracle queries as follows:*
   - *On input* $P^* \in \mathcal{P}$ *for* KGen*, return* $\mathsf{dk}_{P^*} \leftarrow\$ \mathsf{KGen}_1(\mathsf{msk}, P^*)$.
   - *On input* $j \in [n]$ *for* Corr*, return* ek_j.
   - *On input* $(x, m) \in \mathcal{X}_{1,j} \times \mathcal{M}_4$ *for* $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$*,* A *proceeds as follows:*

     Case $j = j_b$*: Run* $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\$ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_j, j)|})$ *and* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\$ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{out}}|}, 1^{|m|})$.

     Case $j \neq j_b$*: Compute* $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.

     Finally, return c_j.
3. *Receive the challenge* $((m_1^0, \ldots, m_n^0), (m_1^1, \ldots, m_n^1), (x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *from* D.
4. *For every* $j \in [n]$*, the adversary* A *proceeds as follows:*

Case $j \in [j_b : r-1]_n^+$*: Compute* $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\$ \mathsf{S}_3(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{in}}|}, 1^{|(\mathsf{sk}_j, j)|})$ *and* $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\$ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_j^{\mathsf{out}}|}, 1^{|m_j^b|})$.

Case $j = r$*: Compute* $c_r^{(v)} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_r^{(v-1)})$ *for* $v \in [n] \setminus [j_b - 1]$ *where* $c_r^{(j_b-1)} = w \leftarrow\$ \mathcal{M}_{2,j_b}$*. Send the challenge* $(\mathbb{V}_r^{\mathsf{in}}, (\mathsf{sk}_r, r))$ *to the challenger and receive the answer* $\widetilde{\mathbb{C}}^*$*. Set* $c_r = (\widetilde{\mathbb{C}}_r^{\mathsf{in}}, \widetilde{\mathbb{C}}_r^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_r^{\mathsf{in}} = \widetilde{\mathbb{C}}^*$, $y_r^{\mathsf{out}} \leftarrow\$ \leftarrow\$ s_4(\lambda)$, *and* $\widetilde{\mathbb{C}}_r^{\mathsf{out}} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_r^{\mathsf{out}}, y_r^{\mathsf{out}}, m_r^b)$.

Case $i < n-2$ *and* $j \in [r+1 : j_b - 1]_n^+$*: Compute* $c_j \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_j, x_j^b, m_j^b)$.

5. *Send the challenge ciphertexts* $(c_1, \ldots, c_n)$ *to* D.
6. *Answer to the incoming oracle queries as in Item* 2.
7. *Return the output of* D.

Let d *be the challenge bit sampled by the challenger. The adversary* A *perfectly simulates the view of* D*. In particular, if* $d = 0$, A *simulates* $\mathbf{H}_{7+i}^{b,0,0}(\lambda)$*. On the other hand, if* $d = 1$, A *simulates* $\mathbf{H}_{7+i}^{b,1,0}(\lambda)$*. Hence,* A *has the same advantage of* D*. This concludes the proof.* □

Claim 39. $\mathbf{H}_{7+i}^{b,1,0}(\lambda) \approx_c \mathbf{H}_{7+i}^{b,1,1}(\lambda)$ *for* $i \in \{0\} \cup [n-2]$.

Proof. *Claim 39 follows by leveraging a similar argument to that of Claim 38.* □

Claim 40. $\mathbf{H}_{7+n-2}^{1-b,1,1}(\lambda) \approx_c \mathbf{H}_{7+n-2}^{b,1,1}(\lambda)$.

Proof. *The distribution of these two experiments does not depend on the bit* b*.* □

By combining Claims 31–40 and conditioned to the event $\mathbf{Validity}_{2,j_0,j_1}$, we conclude that

$$\mathbf{H}_0^b \approx_c \mathbf{H}_1^b \equiv \mathbf{H}_2^{b,0} \approx_c \cdots \approx_c \mathbf{H}_2^{b,q} \approx_c \mathbf{H}_3^b \approx_c \mathbf{H}_4^b \equiv \mathbf{H}_5^{b,0} \approx_c \cdots \approx_c \mathbf{H}_5^{b,q} \equiv \mathbf{H}_6^{b,0} \approx_c \cdots \approx_c \mathbf{H}_6^{b,q} \equiv \mathbf{H}_6^{b,1,1} \approx_c \mathbf{H}_7^{b,0,0} \approx_c \cdots \approx_c \mathbf{H}_{7+n-2}^{b,1,1} \equiv \mathbf{H}_{7+n-2}^{1-b,1,1}.$$

This concludes the proof. □

Lemma 12. *Let* $j_0 \in [n] \setminus \mathcal{Q}_{\mathsf{Corr}}$*. If* PE *is CPA secure without collusions (Definition 8),* PKE_{2,j_0} *is CPA secure (Definition 6),* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then*

$$\left| \mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{(n-1)\text{-CPA-1-iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_{3,j_0} \right] - \frac{1}{2} \right| \leq \mathit{negl}(\lambda).$$

Proof. *Without loss of generality, let* $q = |\mathcal{Q}_{j_0}| \in \mathit{poly}(\lambda)$ *(recall* $j_0 \notin \mathcal{Q}_{\mathsf{Corr}}$*). Consider the hybrid experiments of Lemmas 4 and 11. Formally,*

- *Let* $\mathbf{H}_0^{1,i}(\lambda), \mathbf{H}_1^{1,i}(\lambda), \mathbf{H}_2^{1,i}(\lambda)$ *for* $i \in [n]$ *be the hybrid of Lemma 10 (for the challenge bit* $b = 1$*) except that are conditioned to the event* $\mathbf{Validity}_{3,j_0}$ *(instead of* $\mathbf{Validity}_1$*).*

- *Let* $\mathbf{H}_0^0(\lambda), \mathbf{H}_1^0(\lambda), \mathbf{H}_2^{0,t}(\lambda), \mathbf{H}_3^0(\lambda), \mathbf{H}_4^0(\lambda), \mathbf{H}_5^{0,t}(\lambda), \mathbf{H}_6^{0,t}(\lambda), \mathbf{H}_6^{0,1,1}(\lambda), \mathbf{H}_{7+j}^{0,k_1,k_2}(\lambda)$, *for* $i \in [n]$, $t \in [q]$, $j \in \{0\} \cup [n-2]$, $(k_1, k_2) \times \leftarrow\$^2$, *be the hybrids of Lemma 11 (for the challenge bit* $b = 0$*) except that are conditioned to the event* $\mathbf{Validity}_{3,j_0}$ *(instead of* $\mathbf{Validity}_{2,j_0,j_1}$*).*

In addition, consider the following additional hybrids experiments:

$\mathbf{H}_{7+n-1}^{0,0}$*: Identical to* $\mathbf{H}_{7+n-2}^{0,1,1}$.

$\mathbf{H}_{7+n-1}^{0,i}$ *for* $i \in [q]$*: Same as* $\mathbf{H}_{7+n-1}^{0,i-1}$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_0}, \cdot, \cdot)$*. Formally, on input the* i'*th query* (x, m) *such that* $i' \le i$*, the challenger returns* $c_{j_0} = (\widetilde{\mathbb{C}}_{j_0}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_0}^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_v^{\mathsf{out}} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_{j_0}^{\mathsf{out}}, y_{j_0}^{\mathsf{out}}, m)$ *where* $y_{j_0}^{\mathsf{out}} \leftarrow\$ \leftarrow\$^{s_4(\lambda)}$*. Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_{7+n-1}^{0,0}$.

$\mathbf{H}_{7+n}^{0,0}$*: Identical to* $\mathbf{H}_{7+n-1}^{0,q}$.

$\mathbf{H}_{7+n}^{0,i}$ *for* $i \in [q]$*: Same as* $\mathbf{H}_{7+n}^{0,i-1}$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_0}, \cdot, \cdot)$*. Formally, on input the* i'*th query* (x, m) *such that* $i' \le i$*, the challenger returns* $c_{j_0} = (\widetilde{\mathbb{C}}_{j_0}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_0}^{\mathsf{out}})$ *where* $\widetilde{\mathbb{C}}_{j_0}^{\mathsf{in}} \leftarrow\$ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_{j_0}^{\mathsf{in}}, y_{j_0}^{\mathsf{in}}, (\mathsf{sk}_{j_0}, j_0))$ *where* $y_{j_0}^{\mathsf{in}} \leftarrow\$ \leftarrow\$^{s_3(\lambda)}$*. Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_{7+n}^{0,0}$.

$\mathbf{H}_{7+n+1}^{0,0}$*: Identical to* $\mathbf{H}_{7+n}^{0,q}$.

$\mathbf{H}_{7+n+1}^{0,i}$ *for* $i \in [q]$*: Same as* $\mathbf{H}_{7+n+1}^{0,i-1}$ *except that the challenger changes how it answers to the first* i *queries for oracle* $\mathsf{Enc}(\mathsf{ek}_{j_0}, \cdot, \cdot)$*. On input the* i'*th query* (x, m) *such that* $i' \le i$*, the challenger samples* $(y_{j_0}^{\mathsf{in}}, y_{j_0}^{\mathsf{out}}) \leftarrow\$ \leftarrow\$^{s_3(\lambda)+s_4(\lambda)}$ *and computes* $c_{j_0}^{(0)} \leftarrow\$ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_{j_0}^{\mathsf{in}}, y_{j_0}^{\mathsf{out}}))$ *where* $x_{j_0} = x$*, and* $x_j = x_j^\star$ *for* $j \in [n] \setminus \{j_0\}$*. Finally, the challenger returns* $c_{j_0} = (\widetilde{\mathbb{C}}_{j_0}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_0}^{\mathsf{out}})$ *where* $c_{j_0}^{(v)} \leftarrow\$ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_0}^{(v-1)})$ *for* $v \in [n]$, $\widetilde{\mathbb{C}}_{j_0}^{\mathsf{in}} \leftarrow\$ \leftarrow\$ \mathsf{Obf}_3(1^\lambda, \mathbb{V}_{j_0}^{\mathsf{in}}, y_{j_0}^{\mathsf{in}}, (\mathsf{sk}_{j_0}, j_0))$, $\widetilde{\mathbb{C}}_v^{\mathsf{out}} \leftarrow\$ \mathsf{Obf}_4(1^\lambda, \mathbb{V}_{j_0}^{\mathsf{out}}, y_{j_0}^{\mathsf{out}}, m)$*. Otherwise, on input the* i'*th query* (x, m) *such that* $i' > i$*, the challenger answers as usual, i.e., as defined in* $\mathbf{H}_{7+n+1}^{0,0}$.

Claim 41. $\mathbf{H}_0^0(\lambda) \approx_c \mathbf{H}_{7+n-2}^{0,1,1}(\lambda)$.

Proof. *The proof of Claim 41 is identical to that of Lemma 5 where the challenge bit is* $b = 0$. □

Claim 42. $\mathbf{H}_{7+n-1}^{0,i-1}(\lambda) \approx_c \mathbf{H}_{7+n-1}^{0,i}(\lambda)$ *for* $i \in [q]$.

Proof. *Claim 42 follows by leveraging a similar argument to that of Claim 36.* □

Claim 43. $\mathbf{H}_{7+n}^{0,i-1}(\lambda) \approx_c \mathbf{H}_{7+n}^{0,i}(\lambda)$ *for* $i \in [q]$.

Proof. *Claim 43 follows by leveraging a similar argument to that of Claim 35.* □

Claim 44. $\mathbf{H}^{0,i-1}_{7+n+1}(\lambda) \approx_c \mathbf{H}^{0,i-1}_{7+n+1}(\lambda)$ *for* $i \in [q]$.

Proof. *Claim 44 follows by leveraging a similar argument to that of Claim 32.* □

Claim 45. $\mathbf{H}^{1,0}_{0}(\lambda) \approx_c \mathbf{H}^{1,q}_{2}(\lambda)$.

Proof. *The proof of Claim 45 is identical to that of Lemma 4 where the challenge bit is* $b = 1$. □

Claim 46. $\mathbf{H}^{0,q}_{7+n+1}(\lambda) \equiv \mathbf{H}^{1,q}_{2}(\lambda)$.

Proof. *Claim 46 follows by observing that experiments* $\mathbf{H}^{0,q}_{7+n+1}(\lambda)$ *and* $\mathbf{H}^{1,q}_{2}(\lambda)$ *are identical (and does not depend on the bit b).* □

By combining Claims 41–46 and the fact that $\mathbf{Validity}_{3,j_0,}$ holds, we conclude that

$$\begin{aligned}\mathbf{H}^{0}_{0} &\approx_c \mathbf{H}^{0,1,1}_{7+n-2} \equiv \mathbf{H}^{b,0}_{7+n-1} \approx_c \ldots \approx_c \mathbf{H}^{b,q}_{7+n-1} \equiv \mathbf{H}^{b,0}_{7+n} \approx_c \ldots \approx_c \mathbf{H}^{b,q}_{7+n} \\ &\equiv \mathbf{H}^{b,0}_{7+n+1} \approx_c \ldots \approx_c \mathbf{H}^{b,q}_{7+n+1} \equiv \mathbf{H}^{1,q}_{2} \approx_c \mathbf{H}^{1,0}_{0}.\end{aligned}$$

This concludes the proof. □

Lemma 13. *Let* $j_1 \in [n] \setminus \mathcal{Q}_{\mathsf{Corr}}$. *If* PE *is CPA secure without collusions (Definition 8),* PKE_{2,j_1} *is CPA secure (Definition 6),* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then*

$$\left|\mathbb{P}\left[\mathbf{G}^{(n-1)\text{-}\mathsf{CPA}\text{-}1\text{-}\mathsf{iPE}}_{\Pi,\mathsf{A}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_{4,j_1}\right] - \frac{1}{2}\right| \leq \mathit{negl}(\lambda).$$

Proof. *Lemma 13 follows by using a symmetrical argument to that of Lemma 12.* □

By combining Lemmas 10–13 we conclude that Π is CPA secure in the $(n-1)$-corruptions setting without collusions.

CPA-2-sided security of Π (Theorem 7) As usual, consider the predicate space $\mathcal{P} = \{P(x_1, \ldots, x_n)\}$ of Construction 4 where $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \ldots \wedge P_n(x_n)$. Let $P^* \in \mathcal{P}$ be the only predicate for which the adversary will ask for the decryption key dk_{P^*} during the experiment $\mathbf{G}^{(n-1)\text{-}\mathsf{CPA}\text{-}2\text{-}\mathsf{iPE}}_{\Pi,\mathsf{A}}$ (recall that we prove the security of Construction 4 in the scenario without collusions, i.e., $|\mathcal{Q}_{\mathsf{KGen}}| = 1$). We can leverage a similar argument to that used to prove Theorem 6 for the CPA-2-sided case (see Sect. 5.2.1) in order to rewrite the validity condition of $\mathbf{G}^{(n-1)\text{-}\mathsf{CPA}\text{-}2\text{-}\mathsf{iPE}}_{\Pi,\mathsf{A}}$ (Definition 13) as follows:

Either $\mathbf{Validity_1}$ or $\mathbf{Validity_2}$

where

$\mathbf{Validity}_1$: $\forall j \in [n], \forall i_1 \in [k_1 + 1], \ldots, \forall i_n \in [k_n + 1]$,

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

$\mathbf{Validity}_2$: $\forall j \in [n]$, Either $P_j^*(x_j^0) = P_j^*(x_j^1) = 0$ or $P_j^*(x_j^0) = P_j^*(x_j^1) \wedge m_j^0 = m_j^1$

for $\mathcal{Q}_i^b = \{x_i^{(1,b)}, \ldots, x_i^{(k_i,b)}, x_i^{(k_i+1,b)} = x_i^b\}$ for $i \in [n]$, $b \in \leftarrow\!\!\$$ as defined in Definition 13. Recall that, if $i \notin \mathcal{Q}_{\mathsf{Corr}}$, then $\mathcal{Q}_i^b$ is the ordered list composed of the inputs submitted to the oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$ and the challenge input x_i^b. Otherwise, if $i \in \mathcal{Q}_{\mathsf{Corr}}$, then $\mathcal{Q}_i^b$ is equal to the ith input space $\mathcal{X}_{1,i}$ that, in turn, contains also the challenge input x_i^b. Hence, the CPA-2-sided security of Construction 4 follows by proving the following lemmas.

Lemma 14. *If* PE *is CPA secure without collusions (Definition 8),* SKE *is CPA secure (Definition 4), and* LOBF *is secure (Definition 2), then*

$$\left| \mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{(n-1)\text{-CPA-2-iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_1 \right] - \frac{1}{2} \right| \leq \mathit{negl}(\lambda).$$

Proof. *Note that* $\mathbf{Validity}_1$ *is equivalent to the validity condition of CPA-1-sided security. Hence, the lemma follows by leveraging an identical argument to that of the CPA-1-sided case (Sect. 5.3.1).* □

Lemma 15. *If* PE *is CPA-2-sided secure without collusions (Definition 9) and* LOBF *is secure (Definition 2), then*

$$\left| \mathbb{P}\left[\mathbf{G}_{\Pi,\mathsf{A}}^{(n-1)\text{-CPA-2-iPE}}(\lambda) = 1 \wedge |\mathcal{Q}_{\mathsf{KGen}}| = 1 \middle| \mathbf{Validity}_2 \right] - \frac{1}{2} \right| \leq \mathit{negl}(\lambda).$$

Proof. *Let* $P^* \in \mathcal{Q}_{\mathsf{KGen}}$ *and* $((x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ *be the predicate submitted to the oracle* KGen *and the challenge inputs chosen by the adversary, respectively. Despite* P^* *is chosen adaptively, we assume that the values* $\{z_i\}_{i\in[n]}$ *such that* $P_i^*(x_i^0) = P_i^*(x_i^1) = z_i$ *are known before the challenge phase. Indeed,* $\{z_i\}_{i\in[n]}$ *can be guessed with non-negligible probability since* $n = O(1)$.

Consider the following hybrid experiments:

$\mathbf{H}_0^{b,0}(\lambda)$*: This is exactly the experiment* $\mathbf{G}_{\Pi,\mathsf{A}}^{(n-1)\text{-CPA-2-iPE}}(\lambda)$ *conditioned to the event* $\mathbf{Validity}_2$ *where the challenge bit is b, i.e., the adversary is valid and satisfies* $\mathbf{Validity}_2$.

$\mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [n]$*: Same as* $\mathbf{H}_0^{b,i-1}$*, except that the challenger changes how it computes the challenge ciphertext* c_i *with respect to* z_i*. If* $z_i = 0$ *(i.e.,* $P_i^*(x_i^0) = P_i^*(x_i^1) = 0$*), the value* $c_i^{(0)}$ *is computed as* $c_i^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), 0^{s_3(\lambda)+s_4(\lambda)})$ *where* $x_i = x_i^0$*, and* $x_j = x_j^\star$ *for* $j \in [n]\backslash\{i\}$*. Otherwise, if* $z_i = 1$ *(i.e.,* $P_i^*(x_i^0) = P_i^*(x_i^1) = 1$*), the value* $c_i^{(0)}$ *is computed as* $c_i^{(0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_1, \ldots, x_n), (y_i^{\mathsf{in}}, y_1^{\mathsf{out}}))$ *where* $(y_i^{\mathsf{in}}, y_i^{\mathsf{out}}) \leftarrow\!\!\$\ \leftarrow\!\!\$\ s_3(\lambda)+s_4(\lambda)}$*,* $x_i = x_i^0$*, and*

$x_j = x_j^\star$ *for* $j \in [n]\backslash\{i\}$. *Observe that* $c_i^{(0)}$ *is computed by fixing* $x_i = x_i^0$ *(instead of* $x_i = x_i^b$*), i.e., the input* $(x_1, \ldots, x_n)$ *used to compute the* i*th challenge ciphertext is fixed and does not depend on the challenge bit* b.

$\mathbf{H}_1^{b,0}(\lambda)$: *Identical to* $\mathbf{H}_1^{b,n}(\lambda)$.

$\mathbf{H}_1^{b,i}(\lambda)$ *for* $i \in [n]$: *Same as* $\mathbf{H}_1^{b,i-1}$, *except that the challenger changes how it computes the challenger ciphertext* c_i *with respect to* z_i. *If* $z_i = 0$ *(i.e.,* $P_i^*(x_i^0) = P_i^*(x_i^1) = 0$*), the value* $\widetilde{\mathbb{C}}_i^{\mathsf{out}}$ *of challenge ciphertext* $c_i = (\widetilde{\mathbb{C}}_i^{\mathsf{in}}, \widetilde{\mathbb{C}}_i^{\mathsf{out}})$ *is simulated by the challenger using the simulator of the lockable obfuscation scheme* LOBF_4, *i.e.,* $\widetilde{\mathbb{C}}_i^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{S}_4(1^\lambda, 1^{|\mathbb{V}_i^{\mathsf{out}}|}, 1^{|m_i^b|})$ *where* $\mathbb{V}_i^{\mathsf{out}} = \mathbb{C}^{\mathsf{out}}_{c_i^{(n)}, \mathsf{sk}_i, i}$. *Otherwise, if* $z_i = 1$ *(i.e.,* $P_i^*(x_i^0) = P_i^*(x_i^1) = 1$*), the value* $\widetilde{\mathbb{C}}_i^{\mathsf{out}}$ *is computed as in* $\mathbf{H}_1^{b,0}(\lambda)$.

We can prove that the indistinguishability of the above hybrids by leveraging similar techniques to that of Sects. 5.2.1 *and* 5.3.1.

Claim 47. $\mathbf{H}_0^{b,i-1}(\lambda) \approx_c \mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [n]$.

Proof. *Note that the values* $\{z_i\}_{i\in[n]}$ *(i.e.,* $P_i^*(x_i^0) = P_i^*(x_i^1) = z_i$*), can be correctly guessed with non-negligible probability since* $n = O(1)$. *Conditioned to the above, the claim follows from the CPA-2-sided security of* PE. □

Claim 48. $\mathbf{H}_1^{b,i-1}(\lambda) \approx_c \mathbf{H}_1^{b,i}(\lambda)$ *for* $i \in [n]$.

Proof. *As usual, the values* $\{z_i\}_{i\in[n]}$ *(i.e.,* $P_i^*(x_i^0) = P_i^*(x_i^1) = z_i$*), can be correctly guessed with non-negligible probability since* $n = O(1)$. *Conditioned to the above, the claim follows from the security of the lockable obfuscation scheme* LOBF_4. □

Claim 49. $\mathbf{H}_1^{b,n}(\lambda) \equiv \mathbf{H}_1^{1-b,n}(\lambda)$.

Proof. *The claim follows by leveraging the fact that* $\mathbf{Validity}_2$ *holds (i.e., the adversary satisfies* $\mathbf{Validity}_2$*) and observing that the values* $\{z_i\}_{i\in[n]}$ *(i.e.,* $P_i^*(x_i^0) = P_i^*(x_i^1) = z_i$*), can be correctly guessed with non-negligible probability since* $n = O(1)$. *Conditioned to the above, for every* $i \in [n]$, *if* $P_i^*(x_i^0) = P_i^*(x_i^1) = z_i = 0$ *we have that the* j*th challenge ciphertext* c_j *does not depend on the bit* b. *On the other hand, if* $P_i^*(x_i^0) = P_i^*(x_i^1) = z_i = 1$, *we have that the* j*th challenge ciphertext* c_j *depends on either* m_j^0 *or* m_j^1. *However, by the validity condition* $\mathbf{Validity}_2$ *we have that* $m_j^0 = m_j^1$. *Hence,* $\mathbf{H}_1^{b,n}(\lambda)$ *and* $\mathbf{H}_1^{1-b,n}(\lambda)$ *are identically distributed. This concludes the proof.* □

By combining Claims 47–49 and the fact that $\mathbf{Validity}_2$ holds, we conclude that $\mathbf{H}_0^{0,0} \approx_c \ldots \approx_c \mathbf{H}_0^{0,n} \equiv \mathbf{H}_1^{0,0} \approx_c \ldots \approx_c \mathbf{H}_1^{0,n} \equiv \mathbf{H}_1^{1,n}$. This concludes the proof. □

By leveraging Lemmas 14 and 15, we conclude that Π of Construction 4 is CPA-2-sided secure.

5.4. *Additional Discussion*

On wildcards. Wildcards affect the security guarantee and the expressiveness of the multi-input PE construction depending on the presence of corruptions. In the case of no corruptions (Construction 3), the (single) wildcard can be removed by simply requiring each ith sender not to compute a ciphertext c_i under the corresponding ith wildcard $x_i^\star$, i.e., $\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$ outputs $\perp$ whenever $x_i = x_i^\star$. In other words, we can transform any secure multi-input PE for $P(x_1, \ldots, x_n) = P_1(x_1) \wedge \ldots \wedge P_n(x_n)$ with wildcard $(x_1^\star, \ldots, x_n^\star)$ into a secure multi-input PE for the same predicate $P(x_1, \ldots, x_n)$ without the wildcard. On the other hand, this cannot be done when corruptions are in place (Construction 4). Indeed, if the adversary gets an encryption key ek_i, then it can use the latter to always produce a ciphertext c_i under $x_i^\star$. This means that the adversary can always use ek_i (of the corrupted user) and satisfy the ith predicate P_i (this also affects the security proof of Construction 4. See Sects. 5.3.1, 5.3.1).

On unbounded collusions For completeness, we highlight that if we start from an initial single-input PE scheme PE (of Theorems 6, 7) that is CPA-1-sided secure against *unbounded collusions*, both our Constructions 3 and 4 are CPA-1-sided secure with respect to a *weaker* form of unbounded collusions (but still stronger than no collusions). For the sake of clarity, we focus on our secret-key Construction 3, but the same argument holds for our Construction 4 against corruptions.

In case of no collusions, at the beginning of the proof of Theorem 6 (see Sect. 5.2.1), we show that the adversary's validity condition (of Definition 13) is equivalent to satisfying at least one of the following four conditions: for some $j_0, j_1 \in [n]$,

$$\textbf{Validity}_1 :$$
$$P_1^*(x_1^0) = 0 \wedge \cdots \wedge P_n^*(x_n^0) = 0 \wedge P_1^*(x_1^1) = 0 \wedge \cdots \wedge P_n^*(x_n^1) = 0 \tag{9}$$
$$\textbf{Validity}_{2,j_0,j_1} : \forall x'_{j_0} \in \mathcal{Q}_{j_0}, \forall x'_{j_1} \in \mathcal{Q}_{j_1},$$
$$P_{j_0}^*(x_{j_0}^0) = 0 \wedge P_{j_0}^*(x'_{j_0}) = 0 \wedge P_{j_1}^*(x_{j_1}^1) = 0 \wedge P_{j_1}^*(x'_{j_1}) = 0 \tag{10}$$
$$\textbf{Validity}_{3,j_0} : \forall x'_{j_0} \in \mathcal{Q}_{j_0},$$
$$P_{j_0}^*(x_{j_0}^0) = 0 \wedge P_{j_0}^*(x'_{j_0}) = 0 \wedge P_1^*(x_1^1) = 0 \wedge \cdots \wedge P_n^*(x_n^1) = 0 \tag{11}$$
$$\textbf{Validity}_{4,j_1} : \forall x'_{j_1} \in \mathcal{Q}_{j_1},$$
$$P_1^*(x_1^0) = 0 \wedge \cdots \wedge P_n^*(x_n^0) = 0 \wedge P_{j_1}^*(x_{j_1}^1) = 0 \wedge P_{j_1}^*(x'_{j_1}) = 0 \tag{12}$$

where $P^*(x_1, \ldots, x_n) = (P_1^*(x_1) \wedge \cdots \wedge P_n^*(x_n)) \in \mathcal{Q}_{\mathsf{KGen}}$ is the *single* key generation query submitted by the adversary A, $((x_1^0, \ldots, x_n^0), (x_1^1, \ldots, x_n^1))$ is the adversarial challenge inputs, and $\mathcal{Q}_i$ are the predicate inputs submitted to the encryption oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$ for $i \in [n]$.

When working with CPA-1-sided security against (fully fledged) unbounded collusions, a valid adversary can obtain two decryption keys for P and P' that satisfy Eq. (10) (or Eqs (11), (12)) with respect to two different indexes $j_0, j_1 \in [n]$ and $j_0', j_1' \in [n]$, i.e., $(j_0, j_1) \neq (j_0', j_1')$. When this happens the proof fails since, as we discussed in Sect. 1.2,

our reduction will make an invalid set of queries to the KGen oracle of the single-input PE. However, we observe that the exact same proof of Theorem 5 goes through when we allow A to asking for multiple decryption keys under the restriction that: $\exists j_0, j_1 \in [n]$, $\forall P(x_1, \ldots, x_n) = (P_1(x_1) \wedge \ldots \wedge P_n(x_n)) \in \mathcal{Q}_{\mathsf{KGen}}$, such that either one condition between Eqs. (9)–(12) is satisfied (i.e., the same indexes j_0, j_1 for all predicates $P \in \mathcal{Q}_{\mathsf{KGen}}$).

6. Applications

In this section, we show the applications of our constructions. In Sect. 6.1, we provide the definitions of ME and we show a construction from multi-key PE. In Sect. 6.2, we define CPA-1-sided reusable robust NI-MPC for all-or-nothing functions and we provide a construction from multi-input PE.

6.1. *Matchmaking Encryption from* 2*-Key PE*

Definition of ME. In ME, a trusted authority generates a decryption key for the receiver, associated to an arbitrary policy of his choice. The receiver is able to decrypt the message if and only if a match occurs, i.e. the sender's attribute match the receiver policy, and vice versa. Differently from [10,11], we consider honest senders (i.e., we do not consider authenticity security). Hence, the sender do not need to receive an encryption key from the authority, but can encrypt a message directly with the sender's attribute as an input. Security against malicious senders (i.e., authenticity) can be achieved by relying on similar techniques of [10,11,26], by combining non-interactive zero-knowledge proofs and digital signatures.

Formally, an ME with message space $\mathcal{M}$, sender's policy and attribute spaces $\mathcal{P}_1$ and $\mathcal{U}_1$, receiver's policy and attribute spaces $\mathcal{P}_2$ and $\mathcal{U}_2$ is composed of the following polynomial-time algorithms:

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

$\mathsf{RKGen}(\mathsf{msk}, \rho)$: The randomized receiver-key generator takes as input the master secret key msk, and attributes $\rho \in \mathcal{U}_2$. The algorithm outputs a secret decryption key dk_ρ for attributes ρ.

$\mathsf{PolGen}(\mathsf{msk}, \mathbb{S})$: The randomized receiver policy generator takes as input the master secret key msk, and a policy $\mathbb{S} \in \mathcal{P}_2$. The algorithm outputs a secret decryption key $\mathsf{dk}_\mathbb{S}$ for the circuit $\mathbb{S}$.

$\mathsf{Enc}(\mathsf{mpk}, \sigma, \mathbb{R}, m)$: The randomized encryption algorithm takes as input the master public key mpk, attributes $\sigma \in \mathcal{U}_1$, a policy $\mathbb{R} \in \mathcal{P}_1$, and a message $m \in \mathcal{M}$. The algorithm produces a ciphertext c linked to both σ and $\mathbb{R}$.

$\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c)$: The deterministic decryption algorithm takes as input a secret decryption key dk_ρ for attributes $\rho \in \mathcal{U}_2$, a secret decryption key $\mathsf{dk}_\mathbb{S}$ for a circuit $\mathbb{S} \in \mathcal{P}_2$, and a ciphertext c. The algorithm outputs a message m.

$\mathbf{G}^{\mathsf{CPA\text{-}t\text{-}ME}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$

$(m^0, m^1, \mathbb{R}^0, \mathbb{R}^1, \sigma^0, \sigma^1, \alpha) \leftarrow\!\$\ \mathsf{A}_0^{\mathsf{RKGen}(\mathsf{msk},\cdot),\mathsf{PolGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk})$

$b \leftarrow\!\$\ \{0,1\}, c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{mpk}, \sigma^b, \mathbb{R}^b, m^b)$

$b' \leftarrow\!\$\ \mathsf{A}_1^{\mathsf{RKGen}(\mathsf{msk},\cdot),\mathsf{PolGen}(\mathsf{msk},\cdot)}(1^\lambda, c, \alpha)$

If $(b' = b)$**: return** 1

Else: return 0

Fig. 9. Games defining CPA-t-sided security of ME.

Correctness states that the receiver can obtain the message with overwhelming probability if a match occurs. As for security, we consider the standard definition of ME, namely CPA-1-sided and CPA-2-sided security. Informally, CPA-1-sided security captures the secrecy of the sender's attributes, the sender's policy, and the message when a match does not occur. On the other hand, CPA-2-sided security extends this secrecy even when a match occurs.

Definition 15. (*Correctness of ME*). An ME with message space $\mathcal{M}$, sender's policy and attribute spaces $\mathcal{P}_1$ and $\mathcal{U}_1$, receiver's policy and attribute spaces $\mathcal{P}_2$ and $\mathcal{U}_2$, is correct if $\forall\lambda \in \mathbb{N}$, $\forall m \in \mathcal{M}$, $\forall\sigma \in \mathcal{U}_1$,$\forall\rho \in \mathcal{U}_2$, $\forall\mathbb{R} \in \mathcal{P}_1$, $\forall\mathbb{S} \in \mathcal{P}_2$ such that $\mathbb{S}(\sigma) = 1 \wedge \mathbb{R}(\rho) = 1$:

$$\mathbb{P}\left[\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, \mathsf{Enc}(\mathsf{mpk}, \sigma, \mathbb{R}, m)) = m\right] \geq 1 - \mathsf{negl}(\lambda),$$

where $\forall(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$, $\mathsf{dk}_\rho \leftarrow\!\$\ \mathsf{RKGen}(\mathsf{msk}, \rho)$, and $\mathsf{dk}_\mathbb{S} \leftarrow\!\$\ \mathsf{PolGen}(\mathsf{msk}, \mathbb{S})$. The above probability is taken over the random coins of Setup, RKGen, PolGen, and Enc.

Definition 16. (*CPA-*1*-sided and CPA-*2*-sided security of ME*). Let $t \in [2]$. We say that an ME Π is CPA-t-sided secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}^{\mathsf{CPA\text{-}t\text{-}ME}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 9. Adversary A is called valid if $\forall\rho \in \mathcal{Q}_{\mathsf{RKGen}}$, $\forall\mathbb{S} \in \mathcal{Q}_{\mathsf{PolGen}}$,

- **Case** $t = 1$ **(mismatch only):**

$$\begin{aligned}(\mathbb{R}^0(\rho) = \mathbb{R}^1(\rho) = 0) \vee (\mathbb{S}(\sigma^0) = \mathbb{S}(\sigma^1) = 0)\\ \vee (\mathbb{R}^0(\rho) = \mathbb{S}(\sigma^1) = 0) \vee (\mathbb{R}^1(\rho) = \mathbb{S}(\sigma^0) = 0);\end{aligned} \tag{13}$$

- **Case $t = 2$ (mismatch and match):** Either

$$(\mathbb{R}^0(\rho) = \mathbb{R}^1(\rho) = 0) \vee (\mathbb{S}(\sigma^0) = \mathbb{S}(\sigma^1) = 0)$$
$$\vee (\mathbb{R}^0(\rho) = \mathbb{S}(\sigma^1) = 0) \vee (\mathbb{R}^1(\rho) = \mathbb{S}(\sigma^0) = 0)$$

$$\text{or } (\mathbb{R}^0(\rho) = \mathbb{R}^1(\rho)) \wedge (\mathbb{S}(\sigma^0) = \mathbb{S}(\sigma^1)) \wedge (m^0 = m^1). \tag{14}$$

We stress that CPA-1-sided and CPA-2-sided security reflects the "mismatch condition" and "match condition" of the original work of Ateniese et al. [10, Definition 5]. We chose to change their names to avoid confusion and make the notation consistent with respect to the one of PE. Also, we stress that [10, Definition 5] defines security of ME only in term of CPA-2-sided security (whereas, in this work, we also consider the weaker notion of CPA-1-sided security).

6.1.1. *Construction of ME from 2-Key PE*

Construction 5. *Let* $\mathsf{kPE} = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *be a 2-key PE scheme with message space* $\mathcal{M}$*, input space* $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$*, and predicate space* $4\mathcal{P} = \{P_{\rho,\mathbb{R}}(x_1, x_2)\}_{(\rho,\mathbb{R})\in\mathcal{V}}$ *indexed by* $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ *such that*

$$P_{\rho,\mathbb{R}}(\sigma, \mathbb{S}) = P_\rho(\mathbb{S}) \wedge P_{\mathbb{R}}(\sigma) = \mathbb{S}(\rho) \wedge \mathbb{R}(\sigma),$$

where $\sigma \in \mathcal{X}_1$*,* $\mathbb{S} \in \mathcal{X}_2$*,* $\rho \in \mathcal{V}_1$*, and* $\mathbb{R} \in \mathcal{V}_2$*. We build an ME scheme with message space* $\mathcal{M}$*, sender's policy and attribute spaces* $\mathcal{X}_2$ *and* $\mathcal{X}_1$*, and receiver's policy and attribute spaces* $\mathcal{V}_2$ *and* $\mathcal{V}_1$*, in the following way:*

$\mathsf{Setup}(1^\lambda)$*: Upon input the security parameter* 1^λ*, the randomized setup algorithm outputs* $\mathsf{mpk} = \mathsf{mpk}$ *and* $\mathsf{msk} = (\mathsf{msk}_1, \mathsf{msk}_2)$ *where* $(\mathsf{mpk}, \mathsf{msk}_1, \mathsf{msk}_2) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$.

$\mathsf{RKGen}(\mathsf{msk}, \rho)$*: Upon input the master secret key* $\mathsf{msk} = (\mathsf{msk}_1, \mathsf{msk}_2)$ *and attributes* $\rho \in \mathcal{V}_1$*, the randomized receiver-key generator outputs* $\mathsf{dk}_\rho \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}_1, \rho)$.

$\mathsf{PolGen}(\mathsf{msk}, \mathbb{S})$*: Upon input the master secret key* $\mathsf{msk} = (\mathsf{msk}_1, \mathsf{msk}_2)$ *and a policy* $\mathbb{S} \in \mathcal{V}_2$*, the randomized receiver policy generator outputs* $\mathsf{dk}_\mathbb{S} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}_2, \mathbb{S})$.

$\mathsf{Enc}(\mathsf{mpk}, \sigma, \mathbb{R}, m)$*: Upon input the master public key* mpk*, attributes* $\sigma \in \mathcal{X}_1$*, a policy* $\mathbb{R} \in \mathcal{X}_2$*, and a message* $m \in \mathcal{M}$*, the randomized encryption algorithm computes* $c \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (\sigma, \mathbb{R}), m)$.

$\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c)$*: Upon input a secret decryption key* dk_ρ *for attributes* $\rho \in \mathcal{V}_1$*, a secret decryption key* $\mathsf{dk}_\mathbb{S}$ *for a policy* $\mathbb{S} \in \mathcal{V}_2$*, and a ciphertext* c*, the deterministic decryption algorithm outputs* $m = \mathsf{Dec}_1(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c)$.

Correctness follows from the correctness of kPE. Below, we establish the following result.

Theorem 8. *Let* kPE *be as above.*

1. *If* kPE *is CPA-1-sided secure (Definition 11) then the ME scheme* Π *from Construction 5 is CPA-1-sided secure (Definition 16).*
2. *If* kPE *is CPA-2-sided secure (Definition 11) then the ME scheme* Π *from Construction 5 is CPA-2-sided secure (Definition 16).*

Proof. (CPA-1-sided security of Π) *Suppose there exists a valid PPT adversary* A *with a non-negligible advantage in breaking the CPA-1-sided security of* Π. *We build an adversary* A' *that breaks the CPA-1-sided security of* kPE. A' *is defined as follows:*

1. *Receive* mpk *from the challenger and send it to* A.
2. A' *answers to the incoming oracle queries as follows:*
   - *On input* $\rho \in \mathcal{V}_1$ *for* RKGen, *forward the query* ρ *to* $\mathsf{KGen}(\mathsf{msk}_1, \cdot)$ *and return the answer* dk_ρ.
   - *On input* $\mathbb{R} \in \mathcal{V}_2$ *from* PolGen, *forward the query* $\mathbb{R}$ *to* $\mathsf{KGen}(\mathsf{msk}_2, \cdot)$ *and return the answer* $\mathsf{dk}_\mathbb{R}$.
3. *Receive the challenge* $(m^0, m^1, \mathbb{R}^0, \mathbb{R}^1, \sigma^0, \sigma^1)$ *from* A'. *Send the challenge* (m^0, m^1, x^0, x^1) *where* $x^i = (\sigma^i, \mathbb{S}^i)$ *for* $i \in \leftarrow\!\!\$$. *Forward the challenge ciphertext* c *to* A.
4. *Answer to the incoming oracle queries as in Item* 2.
5. *Return the output of* A.

Let d *be the challenge bit sampled by the challenger.* A' *perfectly simulates the view of* A. *Moreover,* A *is a valid adversary, i.e., it satisfies the mismatch condition of Eq.* (13). *This implies that* $\forall \rho \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)}, \mathbb{R} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_2,\cdot)}$, $P_{\rho,\mathbb{R}}(\sigma^0, \mathbb{S}^0) = \mathbb{S}^0(\rho) \wedge \mathbb{R}(\sigma^0) = 0$ *and* $P_{\rho,\mathbb{R}}(\sigma^1, \mathbb{S}^1) = \mathbb{S}^1(\rho) \wedge \mathbb{R}^1(\sigma) = 0$. *Hence,* A' *is a valid adversary for* $\mathbf{G}^{\text{CPA-1-kPE}}_{\mathsf{kPE},\mathsf{A}'}(\lambda)$. *This concludes the proof.*

(CPA-2-sided security of Π) *The reduction is identical. The only difference is the analysis of the validity of* A'. *Since* A *is a valid adversary with respect to the CPA-2-sided security experiment of* kPE, *i.e., it satisfies Eq.* (14). *This implies that* $\forall \rho \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1,\cdot)}, \mathbb{R} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_2,\cdot)}$, *either* $P_{\rho,\mathbb{R}}(\sigma^0, \mathbb{S}^0) = P_{\rho,\mathbb{R}}(\sigma^1, \mathbb{S}^1) = 0$ *or* $P_{\rho,\mathbb{R}}(\sigma^0, \mathbb{S}^0) = P_{\rho,\mathbb{R}}(\sigma^1, \mathbb{S}^1) \wedge m^0 = m^1$. *Hence,* A' *is a valid adversary for* $\mathbf{G}^{\text{CPA-2-kPE}}_{\mathsf{kPE},\mathsf{A}'}(\lambda)$. *This concludes the proof.* □

6.2. *Non-interactive Multi Party Computation (with Correlated Randomness) from Multi-input PE*

Definition of CPA-1-sided reusable k-robust NI-MPC for all-or-nothing functions. A NI-MPC protocol for a function $f : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to \mathcal{Y}$ is a (non-interactive) protocol between n parties and an evaluator.[26] On initialization, a trusted party executes the setup algorithm $(\mathsf{crs}, \mathsf{ek}_1, \ldots, \mathsf{ek}_n) \leftarrow\!\!\$ \mathsf{Setup}(1^\lambda, f)$. Then, it publishes the common reference string crs and sends the (possibly correlated) encryption keys to the corresponding parties, i.e., the ith party receives the ith encryption key ek_i. After the setup phase, each party, owning an input $v_i \in \mathcal{V}_i$, sends a single message $c_i \leftarrow\!\!\$ \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, v_i)$

[26] Depending on the scenario, the evaluator can be any of the parties running the NI-MPC protocol.

to the evaluator. The latter will be able to compute the output of the function f by executing $f(v_1, \ldots, v_n) = \mathsf{Eval}(\mathsf{crs}, c_1, \ldots, c_n)$. We focus on NI-MPC *without session identifiers*, i.e., the encryption algorithm does not take in input the unique identifier for the current round. Hence, messages computed in different rounds can be interleaved by design (this will affect the security definition of NI-MPC).

Formally, a NI-MPC protocol Π for a function $f : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to \mathcal{Y}$ consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, f)$: Upon input the security parameter 1^λ and a function $f : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to \mathcal{Y}$, the setup algorithm outputs the common reference string crs and n encryption keys $\mathsf{ek}_1, \ldots, \mathsf{ek}_n$.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, v_i)$: Upon input a common reference string crs, an input $v_i \in \mathcal{V}_i$, and an encryption key ek_i, the randomized encryption algorithm outputs a ciphertext c_i.

$\mathsf{Eval}(\mathsf{crs}, c_1, \ldots, c_n)$: Upon input a common reference string crs and n ciphertexts $c_1, \ldots, c_n$, the deterministic evaluation algorithm outputs a value $y \in \mathcal{Y}$.

Correctness states that the evaluation of n ciphertext $(c_1, \ldots, c_n)$, computed over the inputs $(v_1, \ldots, v_n)$, outputs $f(v_1, \ldots, v_n)$

Definition 17. (*Correctness of NI-MPC*). A NI-MPC protocol for a function $f : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to \mathcal{Y}$ is correct if $\forall \lambda \in \mathbb{N}, \forall (v_1, \ldots, v_n) \in \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$, we have:

$$\mathbb{P}\left[\mathsf{Eval}(\mathsf{crs}, c_1, \ldots, c_n) = f(v_1, \ldots, v_n)\right] = 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{crs}, \mathsf{ek}_1, \ldots, \mathsf{ek}_n) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, f)$ and $c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, v_i)$ for $i \in [n]$. The above probability is taken over the random coins of Setup and Enc.

As for security, a k-robust NI-MPC guarantees the secrecy of the inputs of honest parties even in the presence of an adversary that corrupts a set $\mathcal{Q}_{\mathsf{Corr}}$ of k parties (when an adversary corrupts the ith party it obtains its encryption key ek_i and the latter gives to the adversary the ability of producing adversarially chosen messages using ek_i). Following the blueprint of Halevi et al. [32] (see also [14]), this is formalized by an indistinguishability-based definition that states the infeasibility of distinguishing between $(\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_1, v_1^0), \ldots, \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_n, v_n^0))$ and $(\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_1, v_1^1), \ldots, \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_n, v_n^1))$,[27] so long as any interleaving of the honest inputs with any adversarially chosen input $v_i' \in \mathcal{V}_i$, belonging to a corrupted party $i \in \mathcal{Q}_{\mathsf{Corr}}$, produces the same function evaluation. In addition, security of NI-MPC can be formulated in two different settings, named non-reusable and reusable NI-MPC:

- Non-reusable NI-MPC guarantees the secrecy of parties' inputs only if the setup is executed after each round (i.e., a single evaluation $f(v_1, \ldots, v_n)$ per setup is allowed).
- On the other hand, reusable NI-MPC provides a stronger security guarantees allowing parties to use the same setup in multiple rounds. As defined in [32], full-fledged

[27] Simulation-based security of NI-MPC for general functions is impossible. Indeed, simulation-based NI-MPC implies virtual black box (VBB) obfuscation [14,29,32] and the latter is impossible for certain class of circuits/functions [13].

$\mathbf{G}^{\text{ni-mpc}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{crs}, \mathsf{ek}_1, \ldots, \mathsf{ek}_n) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, f_P)$

$((v_1^0, \ldots, v_n^0), (v_1^1, \ldots, v_n^1), \alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{Corr}(\cdot), \{\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, \cdot)\}_{i\in[n]}}(1^\lambda, \mathsf{crs})$

$b \leftarrow\!\!\$\ \{0,1\}, c_1 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_1, v_1^b), \ldots, c_n \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_n, v_n^b)$

$b' \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{Corr}(\cdot), \{\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, \cdot)\}_{i\in[n]}}(1^\lambda, c_1, \ldots, c_n, \alpha)$

If $(b' = b)$**: return** 1

Else: return 0

Fig. 10. Game defining (CPA-1-sided) reusable k-robust security of NI-MPC for all-or-nothing functions and without session identifiers. On input $i \in [n]$, the corruption oracle $\mathsf{Corr}(\cdot)$ returns the ith encryption key ek_i.

reusability NI-MPC makes use of session identifiers in order to block interleaving of messages produced in different rounds. In particular, in each round of computation, the parties compute their messages $c_1, \ldots, c_n$ by attaching to them a unique session identifiers ℓ. Only messages $c_1, \ldots, c_n$ with the same identifier ℓ can be evaluated together yielding $f(v_1, \ldots, v_n) = \mathsf{Eval}(\mathsf{crs}, c_1, \ldots, c_n)$.

We focus on a weaker notion of reusability without session identifiers, specifically tailored for all-or-nothing functions, that allows to re-use the same setup until a certain condition is satisfied. An all-or-nothing function $f_P : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to (\mathcal{M}_1 \times \cdots \times \mathcal{M}_n) \cup \{\perp\}$ returns parties' messages $(m_1, \ldots, m_n) \in \mathcal{M}_1 \times \cdots \times \mathcal{M}_n$ only if a predicate $P(x_1, \ldots, x_n)$ is satisfied, i.e.,

$$f_P(v_1, \ldots, v_n) = \begin{cases} (m_1, \ldots, m_n) & \text{if } P(x_1, \ldots, x_n) = 1 \\ \perp & \text{otherwise} \end{cases} \tag{15}$$

where $v_i = (x_i, m_i) \in \mathcal{V}_i = \mathcal{X}_i \times \mathcal{M}_i$ for $i \in [n]$. We named our weaker notion of reusability *CPA-1-sided reusability* and, in a nutshell, it allows parties to reuse the same setup (without affecting the security of the protocol) so long as f_P evaluates $\perp$ for any combinations of the honest inputs and every input associated to the corrupted parties.[28] This condition resembles the CPA-1-sided security of multi-input PE (Definition 13).

Definition 18. (*CPA-1-sided reusable k-robust security of NI-MPC for all-or-nothing functions*). Let $f_P : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to (\mathcal{M}_1 \times \cdots \times \mathcal{M}_n) \cup \{\perp\}$ be an all-or-nothing function as defined in Eq. (15). We say that a NI-MPC protocol Π for f_P is CPA-1-sided reusable k-robust secure if for any valid PPT adversary $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ we have:

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

where $\mathbf{G}^{\text{ni-mpc}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig. 10. Let $\mathcal{Q}_i = \mathcal{Q}_{\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, \cdot)}$ for $i \in [n] \setminus \mathcal{Q}_{\mathsf{Corr}}$ and $\mathcal{Q}_i = \mathcal{X}_i$ for $i \in \mathcal{Q}_{\mathsf{Corr}}$. Adversary A is called valid if $|\mathcal{Q}_{\mathsf{Corr}}| \leq k$ and $\forall d \in \ \leftarrow\!\!\$$,

[28] We consider every combination of the inputs due to the lack of session identifiers.

$\forall j \in [n], \forall (v'_1, \ldots, v'_n) \in \mathcal{Q}_1 \cup \{v_1^d\} \times \cdots \times \mathcal{Q}_n \cup \{v_n^d\}$, we have that

$$f_P(v'_0, \ldots, v'_{j-1}, v_j^d, v'_{j+1}, \ldots, v'_{n-1}) = \perp.$$

We stress that both the flavors of corruption and challenge selection considered in our Definition 18 are stronger than the one of Halevi et al. [32]. In Definition 18, the adversary can both choose which parties want to corrupt and the challenge adaptively. On the other hand, [32] only covers selective security on both aspects.

Remark 3. (*On the relation between NI-MPC, iO, and null iO*). As note by previous works [14,32], NI-MPC has strong relations with iO. Taking into account full-fledged reusability, indistinguishability-based 0-robust NI-MPC for general functions that supports $n = \mathsf{poly}(\lambda)$ parties implies iO. The construction is reminiscent to that of iO from multi-input functional encryption [29]. Analogously, we can translate the above implications to the setting of CPA-1-sided reusability and null iO (and, in turn WE) [19,31,48], i.e., CPA-1-sided reusable 0-robust NI-MPC for general functions that supports $n = \mathsf{poly}(\lambda)$ parties implies null iO. This shows that nonetheless CPA-1-sided reusability is a weakening of standard reusability, it is non-trivial to achieve for general functions. Moreover, if we consider 1-robustness, we can get rid of both the (CPA-1-sided) reusability and $n = \mathsf{poly}(\lambda)$ parties requirements. In particular, as described in Sect. 1.4, we can build iO (resp. null iO) from indistinguishability-based (resp. CPA-1-sided) non-reusable 1-robust NI-MPC supporting $n = 2$ parties.[29]

6.2.1. *Construction of NI-MPC for all-or-nothing functions from Multi-input PE*

Here, we build a CPA-1-sided reusable k-robust NI-MPC protocol for $f_P : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to (\mathcal{M}_1 \times \cdots \times \mathcal{M}_n) \cup \{\perp\}$ (defined as in Eq. (15)) from any CPA-1-sided secure n-input PE in the k-corruptions setting without collusions.

Construction 6. *Let* $\mathsf{iPE}_1 = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *be a* n*-input PE scheme with message space* $\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_n$*, input space* $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$*, and predicate space* $\mathcal{P}_1 = \{P(x_1, \ldots, x_n)\}$*. Let* $\mathcal{V}_i = \mathcal{X}_i \times \mathcal{M}_i$ *for* $i \in [n]$*. For every* $P \in \mathcal{P}_1$*, we build a NI-MPC protocol for the function* $f_P : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to (\mathcal{M}_1 \times \cdots \times \mathcal{M}_n) \cup \{\perp\}$ *(as defined in Eq.* (15)*) in the following way:*

$\mathsf{Setup}(1^\lambda, f_P)$*: Upon input the security parameter* 1^λ *and a function* f_P*, the randomized setup algorithm computes* $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$ *and* $\mathsf{dk}_P = \mathsf{KGen}_1(\mathsf{msk}, P)$ *where* $P \in \mathcal{P}_1$ *is the predicate defining the function* f_P*. Finally, it returns* $\mathsf{crs} = \mathsf{dk}_P$ *and* $\mathsf{ek}_1, \ldots, \mathsf{ek}_n$*.*

$\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, v_i)$*: Let* $i \in [n]$*. Upon input the common reference string* $\mathsf{crs} = \mathsf{dk}_P$*, the encryption key* ek_i*, and the input* $v_i = (x_i, m_i) \in \mathcal{V}_i$*, the randomized encryption algorithm outputs* $c_i \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{ek}_i, x_i, m_i)$*.*

[29] Non-reusable 1-robust security of NI-MPC means that the honest encryption key ek_i is used only once (i.e., to compute a single message) whereas ek_{1-i} is revealed to the adversary (i.e., the adversary can use it multiple times without breaking the security of the NI-MPC protocol).

$\mathsf{Eval}(\mathsf{crs}, c_1, \ldots, c_n)$: *On input the common reference string* $\mathsf{crs} = \mathsf{dk}_P$ *and* n *ciphertexts* $c_1, \ldots, c_n$*, the evaluation algorithm outputs* $\mathsf{Dec}_1(\mathsf{dk}_P, c_1, \ldots, c_n)$.

Correctness follows from that of the underlying n-input PE iPE_1. In particular, correctness for the case $f_P((x_1, m_1), \ldots, (x_n, m_n)) = \perp$ (i.e., P is not satisfied) can be obtained by extending the iPE_1's correctness to the case of P is not satisfied, i.e., $\mathsf{Dec}(\mathsf{dk}_P, c_1, \ldots, c_n) = \perp$ whenever $P(x_1, \ldots, x_n) = 0$.[30]

Security of Construction 6 is formalized by Theorem 9. By combining Theorems 9 and 6 (and [30]), we obtain a CPA-1-sided reusable 0-robust NI-MPC protocol for $n = \mathsf{poly}(\lambda)$ parties (based on the LWE assumption) for all-or-nothing functions f_P (Eq. (15)) where P is a conjunctions of arbitrary predicates with wildcards. Similarly, by combining Theorems 9 and 7, we obtain a CPA-1-sided reusable $(n-1)$-robust NI-MPC protocol for $n = O(1)$ parties for the same class of functions. Both settings are non-trivial, and they both imply null iO (and WE) in the case of NI-MPC for general functions (see Sects. 1.3 and Remark 3).

Theorem 9. *Let* iPE_1 *as above. If* iPE_1 *is CPA-1-sided secure in the* k*-corruptions setting without collusions (Definition 13), then* Π *of Construction 6 is CPA-1-sided reusable* k*-robust secure (Definition 18).*

Proof. *Suppose there exists a valid PPT adversary* A *with a non-negligible advantage in breaking the partial reusability* k*-robust security of NI-MPC. we build an adversary* A' *that breaks the CPA-1-sided security in the* k*-corruptions setting without collusions of* iPE_1. A' *proceeds as follows:*

1. *Send* P *to the oracle* $\mathsf{KGen}_1(\mathsf{msk}, \cdot)$ *and receive* dk_P.
2. *Send* $\mathsf{crs} = \mathsf{dk}_P$ *to* A.
3. A' *answers the incoming oracle queries as follows:*
   - *On input* $v_i = (x, m) \in \mathcal{V}_i$ *for* $\mathsf{Enc}(\mathsf{crs}, \mathsf{ek}_i, \cdot)$ *where* $i \in [n]$*, forward the query* (x, m) *to* $\mathsf{Enc}_1(\mathsf{ek}_i, \cdot, \cdot)$ *and return the answer* c_i *to* A.
   - *On input* $i \in [n]$ *for* $\mathsf{Corr}(\cdot)$*, forward the query* i *to oracle* $\mathsf{Corr}_1(\cdot)$ *and return the answer* ek_i *to* A.
4. *Receive the challenge* $(v_1^0 = (x_1^0, m_1^0), \ldots, v_n^0 = (x_n^0, m_n^0))$ *and* $(v_1^1 = (x_1^1, m_1^1), \ldots, v_n^1 = (x_n^1, m_n^1))$.
5. *Send* $((m_1^0, \ldots, m_n^0), (x_1^0, \ldots, x_n^0))$ *and* $((m_1^1, \ldots, m_n^1), (x_1^1, \ldots, x_n^1))$ *to the challenger.*
6. *Receive the ciphertexts* $(c_1, \ldots, c_n)$ *and forward them to* A.
7. *Answer to the incoming oracle queries as in Item* 3.
8. *Return the output of* A.

The adversary A' *perfectly simulates the view of* A. *Moreover, by combining* $|\mathcal{Q}_{\mathsf{KGen}_1}| = 1$ *(*A *submits a single query to the* KGen_1 *oracle) and* A*'s validity, we can easily conclude*

[30] Correctness for the case $P(x_1, \ldots, x_n) = 0$ can be seamlessly added to any multi-input PE scheme by applying an efficiently computable and invertible padding $\Phi(\cdot)$ (e.g., $\Phi(m) = m||1||0^\lambda$ where λ is the security parameter) before encrypting the message m_i, i.e., $\mathsf{Enc}(\mathsf{ek}_i, x_i, \Phi(m_i))$. On decryption, the n-input PE scheme will return $\perp$ whenever the padding is invalid.

that A' *is a valid adversary for the experiment* $\mathbf{G}^{k\text{-CPA-1-iPE}}_{\mathsf{iPE},\mathsf{A}'}(\lambda)$ *without collusions. This concludes the proof.* □

Acknowledgements

The first author was supported by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the second and the fourth were supported by project SERICS (PE00000014) and by project PARTHENON (B53D23013000006), under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU; 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 No. 16KISK038, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation).

Funding Open access funding provided by Aarhus Universitet

References

[1] M. Abdalla, F. Benhamouda, R. Gay, From single-input to multi-client inner-product functional encryption, in S.D. Galbraith, S. Moriai, editors, *ASIACRYPT 2019, Part III*. LNCS, vol. 11923 (Springer, Heidelberg, 2019), pp. 552–582

[2] M. Abdalla, F. Benhamouda, M. Kohlweiss, H. Waldner, Decentralizing inner-product functional encryption, in D. Lin, K. Sako, editors, *PKC 2019, Part II*. LNSC, vol. 11443 (Springer, Heidelberg, 2019), pp. 128–157

[3] M. Abdalla, D. Catalano, D. Fiore, R. Gay, B. Ursu, Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings, in H. Shacham, A. Boldyreva, editors, *CRYPTO 2018, Part I*. LNCS, vol. 10991 (Springer, Heidelberg, 2018), pp. 597–627

[4] M. Abdalla, R. Gay, M. Raykova, H. Wee, Multi-input inner-product functional encryption from pairings, in J.-S. Coron, J.B. Nielsen, editors, *EUROCRYPT 2017, Part I*. LNCS, vol. 10210 (Springer, Heidelberg, 2017), pp. 601–626

[5] S. Agrawal, D.M. Freeman, V. Vaikuntanathan, Functional encryption for inner product predicates from learning with errors, in D.H. Lee and X. Wang, editors, *ASIACRYPT 2011*. LNCS, vol. 7073 (Springer, Heidelberg, 2011), pp. 21–40

[6] S. Agrawal, R. Goyal, J. Tomida, Multi-input quadratic functional encryption from pairings, in T. Malkin, C. Peikert, editors, *CRYPTO 2021, Part IV. Virtual Event*. LNCS, vol. 12828 (Springer, Heidelberg, 2021), pp. 208–238

[7] S. Agrawal, R. Goyal, J. Tomida, Multi-input quadratic functional encryption: stronger security, broader functionality, in *TCC 2022* (Springer, 2023), pp. 711–740
[8] S. Agrawal, A. Yadav, S. Yamada, Multi-input attribute based encryption and predicate encryption, in *CRYPTO 2022* (Springer, 2022), pp. 590–621
[9] P. Ananth, A. Jain, Indistinguishability obfuscation from compact functional encryption, in R. Gennaro, M.J.B. Robshaw, editors, *CRYPTO 2015, Part I*. LNCS, vol. 9215 (Springer, Heidelberg, 2015), pp. 308–326
[10] G. Ateniese, D. Francati, D. Nuñez, D. Venturi, Match me if you can: Matchmaking encryption and its applications, in A. Boldyreva, D. Micciancio, editors, *CRYPTO 2019, Part II*. LNCS, vol. 11693 (Springer, Heidelberg, 2019), pp. 701–731
[11] G. Ateniese, D. Francati, D. Nuñez, D. Venturi, Match me if you can: matchmaking encryption and its applications. *J. Cryptol.* **34**(3), 1–50 (2021)
[12] N. Attrapadung, Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more, in P.Q. Nguyen, E. Oswald, editors, *EUROCRYPT 2014*. LNCS, vol. 8441 (Springer, Heidelberg, 2014), pp. 557–577
[13] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, K. Yang. On the (im)possibility of obfuscating programs, in J. Kilian, editor, *CRYPTO 2001*. LNCS, vol. 2139 (Springer, Heidelberg, 2001), pp. 1–18
[14] A. Beimel, A. Gabizon, Y. Ishai, E. Kushilevitz, S. Meldgaard, A. Paskin-Cherniavsky, Non-interactive secure multiparty computation, in J.A. Garay, R. Gennaro, editors, *CRYPTO 2014, Part II*. LNCS, vol. 8617 (Springer, Heidelberg, 2014), pp. 387–404
[15] N. Bitansky, V. Vaikuntanathan, Indistinguishability obfuscation from functional encryption, in V. Guruswami, editor, *56th FOCS* (IEEE Computer Society Press, 2015), pp. 171–190
[16] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, J. Zimmerman, Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation, in E. Oswald, M. Fischlin, editors, *EUROCRYPT 2015, Part II*. LNCS, vol. 9057 (Springer, Heidelberg, 2015), pp. 563–594
[17] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, in S.P. Vadhan, editor, *TCC 2007*. LNCS, vol. 4392 (Springer, Heidelberg, 2007), pp. 535–554
[18] Z. Brakerski, N. Döttling, S. Garg, G. Malavolta, Factoring and pairings are not necessary for iO: circular-secure LWE suffices, in *ICALP 2022* (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022)
[19] Z. Brakerski, A. Jain, I. Komargodski, A. Passelègue, D. Wichs, Non-trivial witness encryption and null-iO from standard assumptions, in D. Catalano, R. De Prisco, editors, *SCN 18*. LNCS, vol. 11035 (Springer, Heidelberg, 2018), pp. 425–441
[20] J. Chen, Y. Li, J. Wen, J. Weng, Identity-based matchmaking encryption from standard assumptions, in *ASIACRYPT 2022* (Springer, 2022)
[21] J. Chotard, E.D. Sans, R. Gay, D.H. Phan, D. Pointcheval, Decentralized multi-client functional encryption for inner product, in T. Peyrin, S. Galbraith, editors, *ASIACRYPT 2018, Part II*. LNCS, vol. 11273 (Springer, Heidelberg, 2018), pp. 703–732
[22] M. Ciampi, L. Siniscalchi, H. Waldner, Multi-client functional encryption for separable functions, in J. Garay, editor, *PKC 2021, Part I*. LNCS, vol. 12710 (Springer, Heidelberg, 2021), pp. 724–753
[23] M. Clear, C. McGoldrick, Multi-identity and multi-key leveled FHE from learning with errors, in R. Gennaro, M.J.B. Robshaw, editors, *CRYPTO 2015, Part II*. LNCS, vol. 9216 (Springer, Heidelberg, 2015), pp. 630–656
[24] P. Datta, T. Okamoto, J. Tomida, Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption, in M. Abdalla, R. Dahab, editors, *PKC 2018, Part II*. LNCS, vol. 10770 (Springer, Heidelberg, 2018), pp. 245–277
[25] D. Francati, D. Friolo, G. Malavolta, D. Venturi, Multi-key and multi-input predicate encryption from learning with errors, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2023), pp. 573–604
[26] D. Francati, A. Guidi, L. Russo, D. Venturi, Identity-based matchmaking encryption without random oracles, in *INDOCRYPT 2021* (Springer, 2021), pp. 415–435
[27] S. Garg, C. Gentry, A. Sahai, B. Waters, Witness encryption and its applications, in D. Boneh, T. Roughgarden, J. Feigenbaum, editors, *45th ACM STOC* (ACM Press, 2013), pp. 467–476

[28] R. Gay, R. Pass, Indistinguishability obfuscation from circular security, in S. Khuller, V.V. Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021* (ACM, 2021), pp. 736–749

[29] S. Goldwasser, S. Dov Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, H.-S. Zhou, Multi-input functional encryption, in P.Q. Nguyen, E. Oswald, editors, *EUROCRYPT 2014*. LNCS, vol. 8441 (Springer, Heidelberg, 2014), pp. 578–602

[30] S. Gorbunov, V. Vaikuntanathan, H. Wee, Predicate encryption for circuits from LWE, in R. Gennaro, M.J.B. Robshaw, editors, *CRYPTO 2015, Part II*. LNCS, vol. 9216 (Springer, Heidelberg, 2015), pp. 503–523

[31] R. Goyal, V. Koppula, B. Waters, Lockable obfuscation, in C. Umans, editor, *58th FOCS* (IEEE Computer Society Press, 2017), pp. 612–621

[32] S. Halevi, Y. Ishai, A. Jain, I. Komargodski, A. Sahai, E. Yogev, Non-interactive multiparty computation without correlated randomness, in T. Takagi, T. Peyrin, editors, *ASIACRYPT 2017, Part III*. LNCS, vol. 10626 (Springer, Heidelberg, 2017), pp. 181–211

[33] S. Halevi, Y. Ishai, A. Jain, E. Kushilevitz, T. Rabin, Secure multiparty computation with general interaction patterns, in M. Sudan, editor, *ITCS 2016* (ACM, 2016), pp. 157–168

[34] S. Halevi, Y. Lindell, B. Pinkas, Secure computation on the web: computing without simultaneous interaction, in P. Rogaway, editor, *CRYPTO 2011*. LNCS, vol. 6841 (Springer, Heidelberg, 2011), pp. 132–150

[35] A. Jain, H. Lin, A. Sahai, Indistinguishability obfuscation from well-founded assumptions, in S. Khuller, V.V. Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021* (ACM, 2021), pp. 60–73

[36] A. Jain, H. Lin, A. Sahai, Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0, in O. Dunkelman, S. Dziembowski, editors, *EUROCRYPT 2022, Part I*. LNCS, vol. 13275 (Springer, Heidelberg, 2022), pp. 670–699

[37] J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, in N.P. Smart, editor, *EUROCRYPT 2008*. LNCS, vol. 4965 (Springer, Heidelberg, 2008), pp. 146–162

[38] A.B. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in H. Gilbert, editor, *EUROCRYPT 2010*. LNCS, vol. 6110 (Springer, Heidelberg, 2010), pp. 62–91

[39] B. Libert, R. Titiu, Multi-client functional encryption for linear functions in the standard model from LWE, in S.D. Galbraith, S. Moriai, editors, *ASIACRYPT 2019, Part III*. LNCS, vol. 11923 (Springer, Heidelberg, 2019), pp. 520–551

[40] A. López-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in H.J. Karloff, T. Pitassi, editors, *44th ACM STOC* (ACM Press, 2012), pp. 1219–1234

[41] P. Mukherjee, D. Wichs, Two round multiparty computation via multi-key FHE. In M. Fischlin, J.-S. Coron, editors, *EUROCRYPT 2016, Part II*. LNCS, vol. 9666 (Springer, Heidelberg, 2016), pp. 735–763

[42] T. Okamoto, K. Takashima, Fully secure functional encryption with general relations from the decisional linear assumption, in T. Rabin, editor, *CRYPTO 2010*. LNCS, vol. 6223 (Springer, Heidelberg, 2010), pp. 191–208

[43] T. Okamoto, K. Takashima, Adaptively attribute-hiding (hierarchical) inner product encryption, in D. Pointcheval, T. Johansson, editors, *EUROCRYPT 2012*. LNCS, vol. 7237 (Springer, Heidelberg, 2012), pp. 591–608

[44] J. Tomida, Tightly secure inner product functional encryption: multi-input and function-hiding constructions, in S.D. Galbraith, S. Moriai, editors, *ASIACRYPT 2019, Part III*. LNCS, vol. 11923 (Springer, Heidelberg, 2019), pp. 459–488

[45] B. Waters, Functional encryption for regular languages, in R. Safavi-Naini, R. Canetti, editors, *CRYPTO 2012*. LNCS, vol. 7417 (Springer, Heidelberg, 2012), pp. 218–235

[46] H. Wee, Dual system encryption via predicate encodings, in Y. Lindell, editor, *TCC 2014*. LNCS, vol. 8349 (Springer, Heidelberg, 2014), pp. 616–637

[47] H. Wee, D. Wichs, Candidate obfuscation via oblivious LWE sampling, in A. Canteaut, F.-X. Standaert, editors, *EUROCRYPT 2021, Part III*. LNCS, vol. 12698 (Springer, Heidelberg, 2021), pp. 127–156

[48] D. Wichs, G. Zirdelis, Obfuscating compute-and-compare programs under LWE, in C. Umans, editor, *58th FOCS* (IEEE Computer Society Press, 2017), pp. 600–611