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

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

[1]Aarhus University
[3]Max Planck Institute for Security and Privacy
[2]Sapienza University of Rome

June 21, 2022

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 multi-key and multi-input PE supporting the conjunction of poly-many arbitrary single-input predicates, assuming the hardness of the standard 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 matchmaking encryption (ME) and non-interactive multi-party computation (NI-MPC).

As a corollary, plugging in our concrete constructions of multi-key and multi-input PE, we obtain the first construction of ME for *arbitrary policies*, as well as NI-MPC with partial reusability for *all-or-nothing* functions and a *constant* number of parties, under the standard LWE assumption. Prior to our work, all of these applications required much heavier tools such as indistinguishability obfuscation or compact functional encryption.

Keywords: predicate encryption, matchmaking encryption, non-interactive MPC, LWE.

Contents

1 Introduction **1**
- 1.1 Our Contributions . . . 1
- 1.2 Related Work . . . 3

2 Technical Overview **4**
- 2.1 Multi-key Predicate Encryption . . . 4
- 2.2 Multi-input Predicate Encryption . . . 5
- 2.3 Applications . . . 13
- 2.4 Relation with Witness Encryption . . . 15

3 Preliminaries **16**
- 3.1 Notation . . . 16
- 3.2 Lockable Obfuscation . . . 16
- 3.3 Symmetric and Public Key Encryption . . . 17
- 3.4 Predicate Encryption . . . 19

4 Multi-key and Multi-input Predicate Encryption **20**
- 4.1 Multi-key PE . . . 21
- 4.2 Multi-input PE . . . 22

5 Constructions **25**
- 5.1 Multi-key PE from PE and Lockable Obfuscation . . . 25
- 5.2 Multi-input PE from PE, Lockable Obfuscation and SKE/PKE . . . 26

A Matchmaking Encryption **34**
- A.1 Security of ME . . . 34
- A.2 ME from 2-Key PE . . . 35

B Relating Multi-key PE and Multi-input PE **37**
- B.1 Multi-input PE in the Hybrid Setting . . . 37
- B.2 Multi-key PE from Multi-input PE . . . 37

C Missing Proofs **39**
- C.1 Proof of Theorem 4 . . . 39
- C.2 Proof of Theorem 5 . . . 43
- C.3 Proof of Theorem 6 . . . 62

1 Introduction

Predicate encryption (PE) [BW07, KSW08, GVW15] 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 $\mathbb{P}$, in such a way that the decryption process reveals the message if and only if the attribute x satisfies the predicate $\mathbb{P}$ (i.e., $\mathbb{P}(x) = 1$). Typically, security of PE requires indistinguishability in the presence of *collusion attacks*, namely, for any pair of attributes (x_0, x_1) and for any pair of messages (m_0, m_1), ciphertexts corresponding to (x_0, m_0) and to (x_1, m_1) are computationally indistinguishable, even for an adversary possessing poly-many decryption keys $\mathsf{dk}_{\mathbb{P}}$, so long as $\mathbb{P}(x_0) = \mathbb{P}(x_1) = 0$ (otherwise it is easy to distinguish).

Recently, there has been a lot of progress in constructing PE supporting expressive predicates under standard assumptions [BW07, KSW08, LOS+10, OT10, OT12, AFV11, Wat12, Wee14, Att14, GVW15]. In particular, Gourbunov, Vaikuntanathan and Wee [GVW15] give a construction of PE for arbitrary predicates under the learning with errors (LWE) assumption.

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) [AFNV19, AFNV21a] for arbitrary policies and non-interactive multi-party computation (NI-MPC) [HLP11] for a constant number of parties. We elaborate on these contributions in the paragraphs below.

Prior to our work, all of the above applications required much stronger assumptions such as indistinguishability obfuscation (iO) [BGI+01]. While recent work made significant progress towards basing iO on standard assumptions [JLS21, JLS22], 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 [GP21, WW21, BDGM22], but they are based on problems whose hardness is less understood.

Multi-key PE. In multi-key PE, we consider an ensemble of predicates $\mathcal{P} = \{\mathbb{P}_v\}$ indexed by a value $v \in \mathcal{V}$ which is uniquely represented as a sequence $v = (v_0, \ldots, v_{n-1}) \in \mathcal{V}_0 \times \ldots \mathcal{V}_{n-1}$. A trusted authority generates decryption keys dk_{v_i} for each $i \in [n]$, with the guarantee that, given the decryption keys $\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}$, the receiver can decrypt successfully the ciphertext c (associated to plaintext m and attributes x), so long as $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x) = 1$.

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 $v_0, \ldots, v_{n-1}$ which correspond to predicates indexed by any value v such that $\mathbb{P}_v(x_0) = \mathbb{P}_v(x_1) = 0$. This yields so-called CPA-1-sided security. The stronger notion of CPA-2-sided security additionally allows for predicates indexed by values v such that $\mathbb{P}_v(x_0) = \mathbb{P}_v(x_1) = 1$, so long as $m_0 = m_1$. These notions mimic the corresponding notions that are already established for standard PE.

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

Our first result is a construction of multi-key PE, from the standard LWE assumption, supporting conjunctions of arbitrary independent predicates, i.e. for predicates of the form $\mathbb{P}(x) = \mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})$, where $x = (x_0, \ldots, x_{n-1})$.

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

Multi-input PE. In multi-input PE, we consider predicates $\mathbb{P}$ with n inputs, i.e. predicates of the form $\mathbb{P}(x_0, \ldots, x_{n-1})$. A trusted authority produces encryption keys ek_i which are associated to the i-th slot of an input for $\mathbb{P}$; namely, given ek_i, a sender can generate a ciphertext c_i which is an encryption of message m_i under attribute x_i. At the same time, the authority can produce a decryption key $\mathsf{dk}_{\mathbb{P}}$ associated to an n-input predicate $\mathbb{P}$, with the guarantee that the receiver can successfully decrypt $c_0, \ldots, c_{n-1}$, and thus obtain $m_0, \ldots, m_{n-1}$, so long as $\mathbb{P}(x_0, \ldots, x_{n-1}) = 1$.

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 $(x_0^0, \ldots, x_{n-1}^0)$ and $(x_0^1, \ldots, x_{n-1}^1)$ and for any pair of sequences of messages $(m_0^0, \ldots, m_{n-1}^0)$ and $(m_0^1, \ldots, m_{n-1}^1)$, ciphertexts $c_0, \ldots, c_{n-1}$ corresponding to either $(x_0^0, m_0^0), \ldots, (x_{n-1}^0, m_{n-1}^0)$ or $(x_0^1, m_0^1), \ldots, (x_{n-1}^1, m_{n-1}^1)$ 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).

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 standard LWE assumption, supporting conjunctions of arbitrary independent predicates *with wildcards*, i.e. for predicates of the form $\mathbb{P}(x_0, \ldots, x_{n-1}) = \mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})$, such that each predicate $\mathbb{P}_i$ has a (public) wildcard input $x_i^\star$ for which $\mathbb{P}_i(x_i^\star) = 1$.

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

Our third result is a construction of multi-input PE, from the standard 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 hardness of LWE, there exists a CPA-1-sided secure multi-input PE scheme supporting conjunctions of $n = O(1)$ arbitrary independent predicates with wildcards, under $n - 1$ adaptive corruptions and without collusions.*

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 independent 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. In particular:

- Two-key PE supporting predicates $\mathbb{P}(x) = \mathbb{P}_0(x_0) \wedge \mathbb{P}_1(x_1)$ implies ME for arbitrary policies. ME is a natural generalization of ABE in which both the sender and the receiver can specify their own attributes and access policies. Previous work showed how to obtain CPA-2-sided secure ME for arbitrary policies using iO [AFNV19, AFNV21a], or for very restricted policies (i.e., for identity matching) using bilinear maps [FGRV21]. In contrast, we obtain ME with CPA-1-sided security for arbitrary policies under the LWE assumption.

- Multi-input PE (with wildcards) supporting predicates $\mathbb{P}(x_0, \ldots, x_{n-1}) = \mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})$ with adaptive corruptions and without collusions implies partially[2] $(n-1)$-robust reusable NI-MPC [BGI+14, HLP11] for the following class of *all-or-nothing* functions:

$$f_{\mathbb{P}}((x_0, m_0), \ldots, (x_{n-1}, m_{n-1})) = \begin{cases} (x_0, m_0, \ldots, x_{n-1}, m_{n-1}) & \text{if } \mathbb{P}(x_0, \ldots, x_{n-1}) = 1 \\ \bot & \text{otherwise} \end{cases}$$

  where $\mathbb{P}$ is a conjunctions of arbitrary independent predicates (with wildcards). NI-MPC allows n parties to evaluate a function f on their inputs using a single round of communication (i.e., each party sends a single message). This is achieved by assuming a trusted setup (that may depend on the function itself) that generates (possibly correlated) strings (e.g., encryption keys) that can be later used by the parties to perform function evaluation. NI-MPC can be defined both in the non-reusable (i.e., the setup is run in every round) and in the reusable setting (i.e., multiple evaluations using the same setup are possible), and intuitively provides the security guarantee that an adversary learns no more than the residual function defined over the choice of the inputs of honest parties. Note that NI-MPC for the above function $f_{\mathbb{P}}$ can, e.g., be used to implement an unanimous voting protocol with message passing, i.e., when all the parties have (secretly) found an unanimous agreement (i.e., $\mathbb{P}_i(x_i) = 1$ for every $i \in [0, n-1]$) the messages (and the votes) are revealed to all the participants.

  Previous works [BGI+14, GGG+14, HIJ+16, HIJ+17] showed that NI-MPC implies iO even if we consider very weak security models, like the non-reusable 1-robust (i.e., one malicious party) setting with $n = 2$, or the reusable 0-robust (i.e., no malicious parties) setting with $n = \mathsf{poly}(\lambda)$. In contrast, we obtain either $(n-1)$-robust reusable NI-MPC with $n = O(1)$ or 0-robust reusable NI-MPC with $n = \mathsf{poly}(\lambda)$—albeit for a restricted class of functions—under the LWE assumption.

1.2 Related Work

Multi-input PE is a special case of multi-input functional encryption (FE) [GGG+14]. It is well known that so-called compact FE (supporting arbitrary functions) implies multi-input FE [AJ15, BV15], which in turn implies iO. Constructions of multi-input FE from standard assumptions, in turn, exist for restricted functions [BLR+15, AGRW17, DOT18, ACF+18, CDG+18, Tom19, ABG19, ABKW19, LT19, AGT21].

Multi-input PE can also be seen as stronger form of multi-input attribute-based encryption (ABE) [BJK+18], the difference being that the attributes are not private in the case of ABE. Previously to our work, all constructions of multi-input ABE required iO.

The multi-input and multi-key settings have also been considered in the context of fully-homomorphic encryption [LTV12, CM15, MW16].

[2] In particular, multiple evaluations of the function with the same setup are possible, so long as $\mathbb{P}(x) = 0$. See Section 2.3 for more details.

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 independent predicates, i.e. predicates of the form:

$$\mathbb{P}(x_0, \ldots, x_{n-1}) = \mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1}). \tag{1}$$

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* [WZ17, GKW17]. Informally, a lockable obfuscation scheme allows to obfuscate a circuit $\mathbb{C}$ under a random lock y together with a message m, in such a way that evaluating the obfuscated circuit returns m if $\mathbb{C}(x) = 1$ (and $\perp$ otherwise). Lockable obfuscation exists under the standard LWE assumption.

Next, 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 is able to handle adaptive corruptions. We provide an high-level overview in the remaining part of this section. For more details, we refer the reader to Section 4 and Section 5.

2.1 Multi-key Predicate Encryption

An n-key PE allows a sender to encrypt a message m under an attribute x, by running $c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, x, m)$. Similarly to single-input PE, a receiver can correctly decrypt c if it has a decryption key for a predicate $\mathbb{P}_v$, within a family $\mathcal{P}$ of predicates indexed by values $v \in \mathcal{V}$, such that $\mathbb{P}_v(x) = 1$. The main difference between single-input PE and n-key PE is that in the latter the receiver must have n independent decryption keys $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}})$ that uniquely represent the predicate $\mathbb{P}_{v_0,\ldots,v_{n-1}}$, 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 $(\mathsf{msk}_0, \ldots, \mathsf{msk}_{n-1})$ are the master secret keys generated during the setup. Hence, once obtained $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}})$ from the authority, the receiver can decrypt the ciphertext c (encrypted under attribute x) by executing $\mathsf{Dec}(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}, c)$. The message is returned if the predicate $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x) = 1$ where $\mathbb{P}_{v_0,\ldots,v_{n-1}}(\cdot)$ is the predicate represented by the combination of the n decryptions keys $\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}$. 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 $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_1})$ 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 $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_1})$ such that $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 0$ (i.e., the adversary cannot decrypt the challenge ciphertext).[3]

As explained above, we focus on conjunctions of arbitrary predicates

$$\mathbb{P}_{v_0,\ldots,v_{n-1}}(x) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0, \ldots, x_{n-1}) = \mathbb{P}_{v_0}(x_0) \wedge \cdots \wedge \mathbb{P}_{v_{n-1}}(x_{n-1})$$

[3] Observe that the decryption keys can be interleaved. For example, starting from $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_i}, \ldots \mathsf{dk}_{v_{n-1}})$ representing the predicate $\mathbb{P}_{v_0,\ldots,v_i,\ldots,v_{n-1}}$, the adversary can ask for an additional i-th decryption key $\mathsf{dk}_{v'_i}$ and rearrange the decryption keys as $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v'_i}, \ldots \mathsf{dk}_{v_{n-1}})$ in order to obtain the tuple representing a different predicate $\mathbb{P}_{v_0,\ldots,v'_i,\ldots,v_{n-1}} \neq \mathbb{P}_{v_0,\ldots,v_i,\ldots,v_{n-1}}$.

as defined in Equation (1); hence, each dk_{v_i} identifies the i-th predicate of the conjunction (and, in turn, any tuple of n decryption keys uniquely identifies the global predicate). We build an n-key PE handling this class of predicates by extending the technique of Goyal *et al.* [GKW17], that uses lockable obfuscation to transform any CPA secure ABE/PE (i.e., secrecy of the message) into a CPA-1-sided secure ABE/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 $i \in [0, n-1]$ be n single-input PE schemes. In a nutshell, our n-key PE scheme $\mathsf{mkPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ works as follows. The setup algorithm Setup simply executes Setup_i of each PE_i and outputs the master public key $\mathsf{mpk} = (\mathsf{mpk}_0, \ldots, \mathsf{mpk}_{n-1})$ and n master secret keys $(\mathsf{msk}_0, \ldots, \mathsf{msk}_{n-1})$. To generate a decryption key $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, v_i)$ (representing the i-th predicate $\mathbb{P}_{v_i}(\cdot)$ of the conjunction), the authority can use the key generation algorithm of the i-th PE, i.e., $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_i, \mathbb{P}_{v_i})$. To encrypt a message m under an input $x = (x_0, \ldots, x_{n-1})$, a sender samples a random lock y and encrypts it n times using $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$, i.e.,

$$c \leftarrow\!\!\$\ \mathsf{Enc}_{n-1}(\mathsf{mpk}_{n-1}, x_{n-1}, \mathsf{Enc}_{n-2}(\mathsf{mpk}_{n-2}, x_{n-2}, \cdots, \mathsf{Enc}_0(\mathsf{mpk}_0, x_0, y))).$$

The final ciphertext of the n-key PE mkPE 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 $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}})$, iteratively decrypts c and returns the last decrypted value, i.e., $y = \mathbb{C}_c(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}) = \mathsf{Dec}_0(\mathsf{dk}_{v_0}, \cdots, \mathsf{Dec}_{n-1}(\mathsf{dk}_{v_{n-1}}, c))$. Decryption is straightforward: the receiver simply executes $\widetilde{\mathbb{C}}$ using its n decryption keys.

The CPA-1-sided security of our construction follows by the CPA-1-sided security of $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$ and by the security of lockable obfuscation. Intuitively, the proof works as follows. In order to be valid, an adversary A cannot hold a tuple of decryption keys $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}})$ such that $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_b) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0^b, \ldots, x_{n-1}^b) = 1$, where $x_b = (x_0^b, \ldots, x_{n-1}^b)$ is the input chosen by A during the challenge phase, and b is the challenge bit. Since $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0^b, \ldots, x_{n-1}^b)$ is a conjunction of arbitrary predicates (see Equation (1)), this implies that there exists an $i \in [0, n-1]$ such that $\mathbb{P}_{v_i}(x_i^b) = 0$ for every i-th decryption key dk_{v_i} obtained by A. We can leverage this observation together with the CPA-1-sided security of PE_i to do a first hybrid in which the challenger computes the i-th layer of the challenge ciphertext as $\mathsf{Enc}_i(\mathsf{mpk}_i, x_i^b, 0 \ldots 0)$. 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 $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$ are CPA-2-sided secure, i.e., $\mathsf{Enc}(\mathsf{mpk}, x_0, m_0)$ and $\mathsf{Enc}(\mathsf{mpk}, x_1, m_1)$ are indistinguishable even when $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 1$ and $m_0 = m_1$. See Section 4.1 and Section 5.1 for more details.

2.2 Multi-input Predicate Encryption

We now turn to the more challenging setting of multi-input PE. 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 $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1})$ and a master secret key msk. Each encryption key ek_i is given to the i-th sender and allows the latter to handle the i-th slot of a multi-input predicate. The i-th party encrypts a message m_i under an input x_i by using its encryption key ek_i, i.e., $c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$. On the other hand, a receiver can use the decryption key $\mathsf{dk}_{\mathbb{P}}$ associated to an n-input predicate $\mathbb{P}$ (recall that $\mathsf{dk}_{\mathbb{P}}$ is generated by the authority via $\mathsf{KGen}(\mathsf{msk}, \mathbb{P})$) to execute $\mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, c_0, \ldots, c_{n-1})$.

Intuitively, the decryption algorithm returns $(m_0, \ldots, m_{n-1})$ when $\mathbb{P}(x_0, \ldots, x_{n-1}) = 1$ where (m_i, x_i) are the message and the input associated to the i-th ciphertext c_i.

The CPA-1-sided security of n-input PE is similar to that of n-key PE, but adapted to the multi-input setting. Informally, an adversary A must not be able to distinguish between $(\mathsf{Enc}(\mathsf{ek}_i, x_i^b, m_i^b))_{i\in[0,n-1]}$ and $(\mathsf{Enc}(\mathsf{ek}_i, x_i^{1-b}, m_i^{1-b}))_{i\in[0,n-1]}$ where $(x_0^0, \ldots, x_{n-1}^0)$, $(x_0^1, \ldots, x_{n-1}^1)$ and $(m_0^0, \ldots, m_{n-1}^0)$, $(m_0^1, \ldots, m_{n-1}^1)$ 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.

2.2.1 Security in the secret key setting

Here, no corruption is allowed and thus the encryption keys are all 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 $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i\in[0,n-1]}$. The latter oracles allow A to generate ciphertexts (associated to the i-th input/sender) on adversarially chosen predicate's inputs and messages. Since these ciphertexts are created independently, the adversary has the power to interleave part of the challenge ciphertext $(c_0^*, \ldots, c_{n-1}^*)$ with the ciphertexts obtained trough the encryption oracles. This has a huge impact on the security of the a n-input PE scheme and on the validity condition that A must satisfy. For example, during the challenge phase, A could choose two vectors of messages $(m_0^0, \ldots, m_{n-1}^0)$ and $(m_0^1, \ldots, m_{n-1}^1)$ and two vectors of predicate inputs $(x_0^0, \ldots, x_{n-1}^0)$ and $(x_0^1, \ldots, x_{n-1}^1)$ such that for every predicate $\mathbb{P}$ (submitted to oracle $\mathsf{KGen}(m, \cdot)$) we have $\mathbb{P}(x_0^0, \ldots, x_{n-1}^0) = \mathbb{P}(x_0^1, \ldots, x_{n-1}^1) = 0$. Although the vector $(c_0^*, \ldots, c_{n-1}^*)$ can not 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 $\mathbb{P}(x_0^0, \ldots, x_i', \ldots x_{n-1}^0) = 1$ and $\mathbb{P}(x_0^1, \ldots, x_i', \ldots x_{n-1}^1) = 0$; (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 $(c_0^*, \ldots, c_i', \ldots, c_{n-1}^*)$. 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 $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i\in[0,n-1]}$. More precisely, for every decryption key $\mathsf{dk}_{\mathbb{P}}$ corresponding to a predicate $\mathbb{P}$, for every vector of ciphertexts obtained by interleaving the challenge ciphertext $(c_0^*, \ldots, c_{n-1}^*)$ with the ciphertexts generated trough any of the n encryption oracles, we must have that $\mathbb{P}$ is not satisfied. This is formalized by the following condition: $\forall b \in \{0,1\}, \forall \mathbb{P} \in \mathcal{Q}_{\mathsf{KGen}}, \forall (x_0', \ldots, x_{n-1}') \in \mathcal{Q}_0 \cup \{x_0^b\} \times \ldots \times \mathcal{Q}_{n-1} \cup \{x_{n-1}^b\}$, $\forall j \in [0, n-1]$ it holds that

$$\mathbb{P}(x_0', \ldots, x_{j-1}', x_j^b, x_{j+1}', \ldots, x_{n-1}') = 0, \tag{2}$$

where $\mathcal{Q}_{\mathsf{KGen}}$ are the queries submitted to oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$, $\mathcal{Q}_i$ are the predicate inputs submitted to oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)$, and $(x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ are the predicate inputs chosen by A during the challenge phase. The formal security definition appears in Section 4.2.

2.2.2 Construction in the secret key setting

We propose a construction of n-input PE for conjunctions of arbitrary predicates (see Equation (1)) *with wildcards* from single-input PE, lockable obfuscation, and SKE. In particular, we

start from single-input PE for arbitrary predicates. Actually, it will suffice that the underlying PE itself supports the predicates $\mathbb{P}(x_0, \ldots, x_{n-1})$ as defined in Equation (1), where we view $(x_0, \ldots, x_{n-1})$ as a single input chosen by the sender. In addition, the predicate must have a wildcard input $(x_0^\star, \ldots, x_{n-1}^\star)$ such that $x_i^\star$ satisfies the i-th predicate of the conjunction, i.e., $\mathbb{P}_i(x_i^\star) = 1$. As we will described next, the $n-1$ subset of wildcards $(x_0^\star, \ldots, x_{i-1}^\star, x_{i+1}^\star, \ldots, x_{n-1}^\star)$ will permit the i-th sender to put a "don't care" placeholder on the slots of the other senders. This will allow the construction to deal with multiple inputs without compromising the evaluation of the predicate.

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

In a nutshell, the i-th secret encryption key has the form $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$ where mpk is the master public key of the single-input PE and k_i for $i \in [0, \ldots, n-1]$ is a secret key for the SKE. In order to encrypt a message m_i under an input x_i, the i-th sender samples a random lock y_i and encrypts (y_i, 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.

$$c_i^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, (x_0^\star, \ldots, x_{i-1}^\star, x_i, x_{i+1}^\star, \ldots, x_{n-1}^\star), (y_i, \mathsf{k}_{i+1})).$$

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 $(c_{i+1}^{(2)}, \ldots, c_{n-1}^{(2)}, c_0^{(2)}, \ldots, c_{i-1}^{(2)})$—note the order of the ciphertexts—and the decryption key $\mathsf{dk}_{\mathbb{P}}$ for $\mathbb{P}(x_0, \ldots, x_{n-1})$, 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.
2. For $c_j^{(2)} \in \{c_{i+1}^{(2)}, \ldots, c_{n-1}^{(2)}, c_0^{(2)}, \ldots, c_{i-1}^{(2)}\}$ do:
   (a) Decrypt $c_j^{(2)}$ using the secret key k, i.e., $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Dec}(\mathsf{k}, c_j^{(2)})$.
   (b) Decrypt $c_j^{(1)}$ using $\mathsf{dk}_{\mathbb{P}}$ 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 $(y_i, \mathsf{k}_{i+1}) = \mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, \mathsf{Dec}(\mathsf{k}, c_i^{(2)}))$, where $c_i^{(2)}$ is the ciphertext hardcoded into the circuit.
4. Return y_i (note that if none of the decryptions fails then y_i is the lock used to obfuscate the circuit).

By the above description, decryption is immediate: Upon input $\{c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})\}$, for $i \in [0, n-1]$, the receiver computes $m_i = \widetilde{\mathbb{C}}_i(c_{i+1}^{(2)}, \ldots, c_{n-1}^{(2)}, c_0^{(2)}, \ldots, c_{i-1}^{(2)}, \mathsf{dk}_{\mathbb{P}})$ where $\mathsf{dk}_{\mathbb{P}}$ is the decryption key of the underlying single-input PE for a predicate $\mathbb{P}(x_0, \ldots, x_{n-1})$. We highlight that the combination of the SKE with the PE wildcards is what allows our construction to correctly implement the predicates of Equation (1). This is because, when $c_i^{(1)}$ correctly decrypts under the key $\mathsf{dk}_{\mathbb{P}}$ (Item 2b), we are guaranteed that $\mathbb{P}_i(x_i) = 1$ (recall that x_i is the input of the

i-th sender). In particular, the latter holds as, in any other slot, the i-th sender has used the wildcards. By repeating this argument, we can conclude that $\mathbb{P}(x_0, \ldots, x_{n-1}) = \mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})$ is satisfied if the execution of $\mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}$ goes as expected. The formal construction is described in Section 5.2.1.

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 $\mathbb{P}^*$ be the predicate submitted to KGen by the adversary. Starting from A's validity condition, for any choice of the challenge bit $b \in \{0, 1\}$, then attacker A must maintain one of the following two conditions:

(i) either $\mathbb{P}_0^*(x_0^b) = \ldots = \mathbb{P}_{n-1}^*(x_{n-1}^b) = 0$ (i.e., all the predicates of the conjunctions are false);

(ii) or (if at least one predicate $\mathbb{P}_i^*$ is satisfied, i.e., $\mathbb{P}_i^*(x_i^b) = 1$) there exists j such that, for every $x_j' \in \mathcal{Q}_j$, it holds that $\mathbb{P}_j^*(x_j^b) = 0 \wedge \mathbb{P}_j^*(x_j') = 0$ where $\mathcal{Q}_j$ are the predicate inputs submitted to the oracle $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$.[4]

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, 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})). 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 $c_{j+1}^{(2)}$ by encrypting the all-zero string (instead of $c_{j+1}^{(1)}$ that, in turn, contains the lock y_{j+1} and the symmetric key 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 $\mathsf{dk}_{\mathbb{P}}$ is crucial in order to get the validity condition (ii), i.e., for every $b \in \{0, 1\}$ there exists a j such that for every predicate (submitted to $\mathsf{KGen}(\mathsf{msk}, \cdot)$) we have $\mathbb{P}_j(x_j^b) = 0$. In fact, in the case of collusions, the adversary can ask for two decryption keys $\mathsf{dk}_{\mathbb{P}}$ and $\mathsf{dk}_{\mathbb{P}'}$ such that for every $b \in \{0, 1\}$:

$$\mathbb{P}_0(x_0^b) = 0 \text{ and } \mathbb{P}_1(x_1^b) = \ldots = \mathbb{P}_{n-1}(x_{n-1}^b) = 1$$
$$\mathbb{P}_0'(x_0^b) = 1 \text{ and } \mathbb{P}_1'(x_1^b) = \ldots = \mathbb{P}_{n-1}'(x_{n-1}^b) = 0.$$

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

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

Lastly, we stress that since we start from a single-input PE supporting conjuctions of arbitrary predicates *with wildcards*, we obtain an n-input PE for conjunctions of arbitrary predicates (see Equation (1)) *with wildcards*. In other words, our n-input PE construction does not support unsatisfiable predicates.

2.2.3 Security under corruptions

Next, let us explain how to define security of multi-input PE in the presence of corruptions. Here, the adversary has the possibility to corrupt a subset of the senders and leak their encryption keys ek_i. We model this by introducing an additional corruption oracle $\mathsf{Corr}(\cdot)$ that, upon input an index $i \in [0, n-1]$, 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 $\mathsf{dk}_{\mathbb{P}}$, for every vector of ciphertexts that can be obtained by interleaving the challenge ciphertext $(c_0^*, \ldots, c_{n-1}^*)$ with both the ciphertexts obtain trough any of the (uncorrupted) encryption oracles and the ones that A may autonomously produce by using the leaked encryption keys (trough oracle $\mathsf{Corr}(\cdot)$), we have that $\mathbb{P}$ is not satisfied. Hence, the validity condition is identical to that of the secret key setting (see Equation (2)), except that:

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

See Section 4.2 for the formal definition.

2.2.4 A simple attack

Before explaining our construction in details, let us show why the previous construction is not secure under corruptions. For simplicity, we focus on the 2-input setting. Suppose an adversary A has a single decryption key $\mathsf{dk}_{\mathbb{P}}$ for $\mathbb{P}(x_0, x_1) = \mathbb{P}_0(x_0) \wedge \mathbb{P}_1(x_1)$ and a vector of ciphertexts $(c_0^*, c_1^*) = ((\widetilde{\mathbb{C}}_0, c_0^{(2)}), (\widetilde{\mathbb{C}}_1, c_1^{(2)}))$ encrypted under the predicate input (x_0, x_1) such that $\mathbb{P}_0(x_0) = 0$ and $\mathbb{P}_1(x_1) = 1$. Note that this ciphertext should not decrypt under $\mathsf{dk}_{\mathbb{P}}$, since the conjunction of $\mathbb{P}_0$ and $\mathbb{P}_1$ evaluates to 0. If A can obtain ek_1, then it can easily determine the message m_1. Indeed, once A gets $\mathsf{ek}_1 = (\mathsf{mpk}, \mathsf{k}_1, \mathsf{k}_0)$, it can compute a malicious ciphertext $\widetilde{c}_0^{(1)}$ (using the single-input PE) by encrypting $(\widetilde{y}, \mathsf{k}_1)$ (where $\widetilde{y}$ is a random lock) under the predicate input composed by all wildcards $(x_0^\star, x_1^\star)$. Then, it can compute $\widetilde{c}_0^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{k}_0, \widetilde{c}_0^{(2)})$ and execute $\widetilde{\mathbb{C}}_1(\widetilde{c}_0^{(2)}, \mathsf{dk}_{\mathbb{P}})$ to get m_1. Note that by definition the execution of $\widetilde{\mathbb{C}}_1$ outputs the correct message, since $\mathbb{P}_0(x_0^\star) \wedge \mathbb{P}_1(x_1) = 1$ and $\widetilde{c}_0^{(2)}$ contains the correct secret encryption key k_1, allowing the circuit to correctly end the computation. Also, note that this attack does not violate the validity condition. This is because $\mathbb{P}_0(x_0) = 0$, and A does not use the oracle $\mathsf{Enc}(\mathsf{ek}_0, \cdot, \cdot)$ at all. Hence, any interleaving of the ciphertexts will involve the predicate input x_0 that, in turn, will make the conjunction $\mathbb{P}(x_0, x_1') = \mathbb{P}_0(x_0) \wedge \mathbb{P}_1(x_1')$ unsatisfied for every choice of the input predicate x_1'.

In light of the above attack, we can identify what we need to do in order to extend our techniques to handle corruptions:

- First, following the proof of the previous construction, it is important to hide the (plain) single-input PE ciphertext that a particular sender produces (e.g., in the secret key setting we re-encrypt $c_i^{(1)}$ using SKE). As we have described for the secret key setting, this allows us to claim that everything remains hidden whenever one of the predicate $\mathbb{P}_i$ of the conjunction is not satisfied (even if a different $\mathbb{P}_j$ is satisfied).[5]

- Second, the leakage of one (or more) encryption keys should not allow to produce a malicious ciphertext on behalf of the uncorrupted senders (or simply decrypt the ciphertexts of other parties). Otherwise, the attacker can follow a strategy similar to the one above to break security.

2.2.5 Construction under corruptions

In order to achieve the above properties, we propose a new technique based on nested (lockable obfuscated) circuits that can be executed one inside the other. This technique permits to make available secret information (e.g., secret keys) *only* during nested execution. For the sake of clarity, we first present our approach for the case of two inputs. As an initial attempt to deal with corruptions, we replace the SKE in our previous construction with a PKE, so that each encryption key ek_i is now composed of $(\mathsf{mpk}, \mathsf{sk}_0, \mathsf{pk}_0, \mathsf{pk}_1)$ where $(\mathsf{sk}_i, \mathsf{pk}_i)$ is a secret/public key pair. Each $(\mathsf{sk}_i, \mathsf{pk}_i)$ is associated to the i-th sender (indeed, note that ek_i contains also the secret key sk_i). From the perspective of the first sender, in order to encrypt a message m_0 under the input x_0, it samples two random locks $(y_0^{\mathsf{in}}, y_0^{\mathsf{out}})$ and encrypts them (using the single-input PE) as before using the wildcard $x_1^\star$, i.e., $c_0^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, (x_0, x_1^\star), (y_0^{\mathsf{in}}, y_0^{\mathsf{out}}))$. At this point, the PE ciphertext $c_0^{(-1)}$ is re-encrypted twice using pk_0 and pk_1, i.e., $c_0^{(i)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}_i, c_0^{(i-1)})$ for $i \in \{0, 1\}$. Intuitively, the two layers of PKE have the role of hiding the PE ciphertexts (that in turn contains the locks) even when the adversary leaks all encryption keys except one. The final ciphertext is composed by the two obfuscations $\widetilde{\mathbb{C}}_0^{\mathsf{out}}$, $\widetilde{\mathbb{C}}_1^{\mathsf{in}}$ of the circuits $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(1)}}$, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_0, c_0^{(1)}}$, respectively. The former is obfuscated under the lock y_0^{out} and message m_0, whereas the latter is obfuscated under the lock y_0^{in} and message sk_0. The ciphertext produced by the second sender, is identical, except that it uses sk_1 (instead of sk_0) and that $c_1^{(-1)}$ is computed using the predicate input $(x_0^\star, x_1)$ (instead of $(x_0, x_1^\star)$).

The crux of our nesting technique comes from the definition of the circuits $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_i, c_i^{(1)}}$. $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_i, c_i^{(1)}}$. More precisely, the outer circuit $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(1)}}$ will take as input the obfuscation $\widetilde{\mathbb{C}}_1^{\mathsf{in}}$ of the inner circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(1)}}$ and a decryption key $\mathsf{dk}_{\mathbb{P}}$. Then, in order to securely check the conjunction inside the lockable obfuscation, $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(1)}}$ will execute $\widetilde{\mathbb{C}}_1^{\mathsf{in}}(\mathsf{sk}_0, \mathsf{dk}_{\mathbb{P}})$. At this point, $\widetilde{\mathbb{C}}_1^{\mathsf{in}}$ has everything it needs to check the satisfiability of $\mathbb{P}_1(\cdot)$. It removes the PKE layers from $c_1^{(1)}$ by computing $c_1^{(-1)} = \mathsf{Dec}(\mathsf{sk}_1, \mathsf{Dec}(\mathsf{sk}_0, c_1^{(1)}))$. Then, it decrypts the PE ciphertext $(y_1^{\mathsf{in}}, y_1^{\mathsf{out}}) = \mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, c_1^{(-1)})$—observe that the decryption succeeds if $\mathbb{P}_1(x_1) = 1$—and returns y_1^{in}. By correctness of lockable obfuscation, if the computation of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(1)}}(\mathsf{sk}_0, \mathsf{dk}_{\mathbb{P}})$ goes as intended, then $\widetilde{\mathbb{C}}_1^{\mathsf{in}}(\mathsf{sk}_0, \mathsf{dk}_{\mathbb{P}})$ will output sk_1 (the message attached to the obfuscation). Once obtained sk_1, the computation of $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(1)}}$ can continue and perform a similar computation to check the satisfiability of $\mathbb{P}_0(\cdot)$ except that, if the PE ciphertext $c_0^{(-1)}$ decrypts correctly, it returns y_0^{out}. If all the decryptions (performed by $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(1)}}$ and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(1)}}$) succeed, the execution of the obfuscation $\widetilde{\mathbb{C}}_0^{\mathsf{out}}$ of $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(1)}}$

[5] The secret key construction achieves this by linking the multiple PE ciphertexts using a SKE and including the secret key k_{i+1} into the PE ciphertext.

will output m_0. As symmetrical argument holds for $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1,c_1^{(1)}}$ and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_0,c_0^{(1)}}$, releasing m_1.

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 $\mathbb{P}_i(\cdot) = 0$ (unless the adversary invoke the oracle $\mathsf{Corr}(i)$). This is reflected by the proof technique that is sketched below.

Let $\mathsf{dk}_{\mathbb{P}^*}$ be the decryption key obtained by A for the predicate $\mathbb{P}^*(\cdot,\cdot) = \mathbb{P}^*_0(\cdot) \wedge \mathbb{P}^*_1(\cdot)$, and let $\mathcal{Q}_{\mathsf{Corr}}$ be the queries submitted to the corruption oracle. Starting from the validity condition, we can infer that for any choice of the challenge bit $b \in \{0,1\}$ we have:

(i) either $\mathbb{P}^*_0(x^b_0) = \mathbb{P}^*_1(x^b_1) = 0$;

(ii) or (i.e., there exists an $i \in \{0,1\}$ such that predicate $\mathbb{P}_i$ is satisfied) $1-i \notin \mathcal{Q}_{\mathsf{Corr}}$ and, for every $x'_{1-i} \in \mathcal{Q}_{1-i}$, $\mathbb{P}^*_{1-i}(x^b_{1-i}) = 0 \wedge \mathbb{P}^*_{1-i}(x'_{1-i}) = 0$. Observe that this second condition holds because of the following:

- If there is $x'_{1-i} \in \mathcal{Q}_{1-i}$ such that $\mathbb{P}^*_{1-i}(x'_{1-i}) = 1$, A can use the corresponding ciphertext to decrypt the i-th part of the challenge ciphertext since $\mathbb{P}^*_i(x^b_i) = 1$.
- If $1-i \in \mathcal{Q}_{1-i}$, A can simply use ek_{1-i} to encrypt a random message under the wildcard $x^\star_{1-i}$ (that always exists by design of our construction) and, again, decrypt the i-th part of the challenge ciphertext.

By leveraging the above two conditions, the security of our scheme follows by using a similar argument to that of the secret key setting. In particular, when the first condition is satisfied, we can show that the locks $(y^{\mathsf{in}}_0, y^{\mathsf{out}}_0)$ and $(y^{\mathsf{in}}_1, y^{\mathsf{out}}_1)$ (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 $(\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0,c_0^{(1)}}, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_0,c_0^{(1)}})$, $(\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_1,c_1^{(1)}}, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1,c_1^{(1)}})$, 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 $\mathsf{Enc}(\mathsf{ek}_{1-i},\cdot,\cdot)$ computes $c^{(-1)}_{1-i}$ by encrypting the all-zero string (instead of $(y^{\mathsf{in}}_{1-i}, y^{\mathsf{out}}_{1-i})$). Then, we can use the security of lockable obfuscation to transition to another hybrid in which $\mathsf{Enc}(\mathsf{ek}_{1-i},\cdot,\cdot)$ simulates all the obfuscations. At this point, the secret key sk_{1-i} of the uncorrupted $(1-i)$-th sender is not used anymore (recall that $1-i \notin \mathcal{Q}_{1-i}$). Hence, we can leverage the security of the PKE to remove the locks $(y^{\mathsf{in}}_i, y^{\mathsf{out}}_i)$ chosen by the i-th sender (whose encrypted key ek_i has been leaked to A, i.e., $i \in \mathcal{Q}_{\mathsf{Corr}}$). In more details, we do another hybrid in which the $(1-i)$-th PKE layer $c^{(1-i)}_i$ of the challenge ciphertext is an encryption of zeroes (instead of $c^{(1-i-1)}_i$ that, in turn, encrypts the locks $(y^{\mathsf{in}}_i, y^{\mathsf{out}}_i)$). After this hybrid, we can again use the security of lockable obfuscation to simulate all the obfuscations (and the corresponding attached messages) that compose the i-th component of the ciphertext. The distribution of this last hybrid does not depend on the challenge bit b since all the ciphertexts are simulated by the simulator of the lockable obfuscation scheme.

To sum up, we can observe that encrypting $c^{(-1)}_i$ (the PE ciphertext that contains the locks) with the public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$ 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.

2.2.6 Generalizing the nesting technique to $(n \geq 2)$ inputs

By carefully modifying the definition of the outer and inner circuits, we can generalize the above technique to the case of $n \geq 2$. 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 $(\mathsf{mpk}, \mathsf{sk}_i, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$.
- To compute the i-th encryption of (x_i, m_i), the sender computes the initial PE ciphertext as $c_i^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, (x_0^\star, \ldots, x_i, \ldots, x_{n-1}^\star), (y_i^{\mathsf{in}}, y_i^{\mathsf{out}}))$. Then, it re-encrypts n times the ciphertext $c_i^{(-1)}$ using $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$, i.e., $c_i^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}_v, c_i^{(v-1)})$ for $i \in [0, n-1]$. 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 $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_i, c_i^{(n-1)}}$ and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_i, c_i^{(n-1)}}$.

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 $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(n-1)}}$ and of the inner circuits $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(n-1)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}})$ generated by the corresponding senders. The remaining circuits are defined similarly. First off, the input space of these circuits is a follows:

- $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(n-1)}}$ takes as input the $n-1$ obfuscations of the circuits $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(n-1)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}})$ and a decryption $\mathsf{dk}_{\mathbb{P}}$. These obfuscations are the inner circuits that needs to be executed in order to return the message m_0 attached to the obfuscation of $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(n-1)}}$.
- On the other hand, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_i, c_i^{(n-1)}}$, for $i \in [n-1]$, takes as input a tuple of n secret keys $(\mathsf{sk}_0, \ldots, \mathsf{sk}_n)$ (where some can be set to $\perp$), a decryption key $\mathsf{dk}_{\mathbb{P}}$, and the obfuscations of $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{i+1}, c_{i+1}^{(n-1)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}})$. 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_0 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 $\mathbb{C}^{\mathsf{out}}_{\mathsf{sk}_0, c_0^{(n-1)}}$ starts the nested execution by invoking the obfuscation of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(n-1)}}$ upon input $(\mathsf{sk}_0, \perp, \ldots, \perp)$, $\mathsf{dk}_{\mathbb{P}}$, and the remaining obfuscations of $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(n-1)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}})$. In turn, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_1, c_1^{(n-1)}}$ will do a similar thing: It executes the next obfuscated circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_2, c_2^{(n-1)}}$ upon input $(\mathsf{sk}_0, \mathsf{sk}_1, \perp, \ldots, \perp)$, $\mathsf{dk}_{\mathbb{P}}$, and the remaining obfuscations $(\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_3, c_3^{(n-1)}}, \ldots, \mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}})$. This process is repeated until $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}}$ is executed upon input $(\mathsf{sk}_0, \ldots, \mathsf{sk}_{n-2}, \perp)$ and $\mathsf{dk}_{\mathbb{P}}$. At this point, all the secret keys are know (observe that sk_{n-1} is hardcoded). From $c_{n-1}^{(n-1)}$, we can remove the n PKE layers, decrypt the PE ciphertext and, in turn, return y_{n-1}^{in} if the PE ciphertext decrypts correctly (i.e., $\mathbb{P}_{n-1}(\cdot)$ is satisfied). Once $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}}$ terminates, the secret key sk_{n-1} is released and $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-2}, c_{n-2}^{(n-1)}}$ performs the computation required to check if $\mathbb{P}_{n-2}(\cdot)$ is satisfied. Indeed, $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-2}, c_{n-2}^{(n-1)}}$ has been executed on input $(\mathsf{sk}_0, \ldots, \mathsf{sk}_{n-3}, \perp, \perp)$, it has sk_{n-2} harcoded, and the execution of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}}$ has released sk_{n-1}. Hence, after the correct termination of $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1}, c_{n-1}^{(n-1)}}$, all secret keys are known.

It may seems that this argument can be iterated. However, there is a problem. Even if $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-2}, c_{n-2}^{(n-1)}}$ correctly terminates, the circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-3}, c_{n-3}^{(n-1)}}$ that invokes it does not have access to the secret key sk_{n-1}. This is because the latter circuit receives as input $(\mathsf{sk}_0, \ldots, \mathsf{sk}_{n-4}, \perp, \perp, \perp)$, it has sk_{n-3} hardcoded, and the circuit $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-2}, c_{n-2}^{(n-1)}}$ has returned sk_{n-2}. As a consequence,

$\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-3},c_{n-3}^{(n-1)}}$ must re-run $\mathbb{C}^{\mathsf{in}}_{\mathsf{sk}_{n-1},c_{n-1}^{(n-1)}}$ on input $(\mathsf{sk}_0,\ldots,\mathsf{sk}_{n-2},\bot)$ in order to get sk_{n-1} 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 Section 5.2.2.

2.3 Applications

We now explain how to apply multi-key and multi-input PE supporting conjunctions of predicates to obtain interesting cryptographic applications.

2.3.1 Matchmaking Encryption

Matchmaking encryption (ME) [AFNV19, AFNV21a] allows a sender to encrypt a message m under some attributes σ and a policy $\mathbb{R}$. On the other hand, the receive 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 that:

- In case of a mismatch, nothing is leaked except the fact that a match did not occur (this is reminiscent to the definition of CPA-1-sided security for ABE and PE).
- Additionally, in case of a match, nothing is leaked except for the message and the fact that a match occurred (this is reminiscent to the definition of CPA-2-sided security for ABE and PE).

Multi-key PE is a direct generalization of ME: 2-key PE for conjunctions $\mathbb{P}_{v_0,v_1}(\cdot,\cdot)=\mathbb{P}_{v_0}(\cdot)\wedge\mathbb{P}_{v_1}(\cdot)$ (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 encryption algorithm of the 2-key PE scheme, i.e., $c\leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk},(x_0,x_1),m)$ where $x_0=\sigma$ and $x_1=\mathbb{R}$. Analogously, the ME decryption keys dk_ρ and $\mathsf{dk}_\mathbb{S}$ correspond to the decryption keys dk_{v_1} and dk_{v_0} of the 2-key PE scheme where $v_0=\mathbb{S}$ and $v_1=\rho$. By setting $\mathbb{P}_{v_0,v_1}(x_0,x_1)=\mathbb{P}_{\mathbb{S},\rho}(\sigma,\mathbb{R})=\mathbb{P}_\sigma(\mathbb{S})\wedge\mathbb{P}_\mathbb{R}(\rho)=\mathbb{S}(\sigma)\wedge\mathbb{R}(\rho)$, 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 first construction of ME for arbitrary policies with CPA-1-sided security (i.e., mismatch security) from LWE. We provide more details in Appendix A.

Previous work [AFNV19, AFNV21a] construct ME for arbitrary policies from much stronger assumptions such as 2-input FE (with one secret key and one public key) and simulation-based randomized FE. For completeness, we highlight that $(n+1)$-input PE supporting arbitrary predicates and tolerating 1 corruption implies n-key PE (see Section 4.1 and Appendix B). 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).

2.3.2 Non-Interactive MPC

Non-interactive MPC (NI-MPC) [BGI+14, HLP11] allows n parties to evaluate a function f on their inputs using a single round of communication (i.e., each party sends a single message). This is achieved by assuming a trusted setup (that may depend on the function itself) that

generates (possibly correlated) strings (e.g., encryption keys) 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 (the reusable definition makes use of session identifiers in order to avoid that an adversary can interleave messages from different rounds). Both non-reusable and reusable NI-MPC provide the same security guarantee: an adversary A learns no more than the residual function defined over the choice of the inputs of honest parties. For instance, in the three-party setting where only the third party is malicious (i.e., corrupted by the adversary), NI-MPC guarantees that the malicious third party cannot learn any information about x or y except from what it can infer from $f(x, y, \cdot)$, where x and y are the inputs of the honest parties.

As mentioned by several works [BGI+14, GGG+14, HIJ+16, HIJ+17], NI-MPC implies iO even if we consider very weak security models. 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 $c_0 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_0, \mathbb{C})$, and outputting $\widetilde{\mathbb{C}} = (c_0, \mathsf{ek}_1)$ where $\mathsf{ek}_0, \mathsf{ek}_1$ are the key material required to encode the inputs of the NI-MPC (note that 1-robustness is necessary since we reveal ek_1). To evaluate the obfuscated circuit, the evaluator only needs to compute $c_1 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_1, x)$ 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 $\mathsf{poly}(\lambda)$ parties implies iO. In this case, iO can be built using a similar construction to that of iO from secret-key multi-input FE [GGG+14].

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

$$f_{\mathbb{P}}((x_0, m_0), \ldots, (x_{n-1}, m_{n-1})) = \begin{cases} (x_0, m_0, \ldots, x_{n-1}, m_{n-1}) & \text{if } \mathbb{P}(x_0, \ldots, x_{n-1}) = 1 \\ \bot & \text{otherwise} \end{cases}$$

where $\mathbb{P}(x_0, \ldots, x_1)$ is a conjunctions of arbitrary independent predicates (with wildcards) as defined in Equation (1). The construction is intuitive. At setup, simply distribute $(\mathsf{dk}_{\mathbb{P}}, \mathsf{ek}_i)$ to the i-th party where $(\mathsf{msk}, \mathsf{ek}_0, \ldots \mathsf{ek}_{n-1}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and $\mathsf{dk}_{\mathbb{P}} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P})$. During evaluation, each party can send the message $c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$ and compute $\mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, c_0, \ldots, c_{n-1})$ to evaluate the function $f_{\mathbb{P}}((x_0, m_0), \ldots, (x_{n-1}, m_{n-1}))$. Security of this NI-MPC for $f_{\mathbb{P}}$ follows readily from CPA-1-sided security of n-input PE under $n-1$ corruptions.

We stress that $f_{\mathbb{P}}$ returns the inputs $(x_0, \ldots, x_1)$ of the parties because the underlying PE satisfies only CPA-1-sided security, that does not guarantee anything about the secrecy of the inputs whenever $\mathbb{P}(x_0, \ldots, x_1) = 1$. Similarly, by leveraging our n-input PE (for $n = \mathsf{poly}(\lambda)$) in the secret-key setting, we can build a 0-robust reusable NI-MPC for a polynomial number of parties (for the same class of predicates), whose security follows directly from CPA-1-sided security of the underlying n-input PE.

Lastly, we highlight that the class of predicates supported by our n-input PE constructions is not expressive enough to seamlessly support session identifiers. For this reason, our NI-MPC protocols only satisfy a form of *partial reusability* which is weaker than fully-fledged reusability (but still significantly stronger than non-reusability). More precisely, partial reusability means

that the adversary does not learn anything except from what it can infer from the multiple residual functions, each one defined with respect to an arbitrary combination (obtained trough an interleaving of messages) of the honest parties's inputs obtained in different rounds. For instance, in the three-party setting where the third party is malicious, this translates into the following guarantee: the adversary does not learn anything except the set of residual functions $\{f(x_0, x_1, \cdot)\}$ for every combination of inputs x_0 and x_1 obtained by interacting in different rounds with the first and second honest party, respectively. Although partial reusability is weaker than fully-fledged reusability, it is non-trivial to achieve since, in the setting of NI-MPC for general functions, it already implies iO, i.e., 1-robust partially-reusable NI-MPC for $n = 2$ parties implies iO.

2.4 Relation with Witness Encryption

In the following we recall the notion of witness encryption witness encryption (WE) [GGSW13] and we discuss its relation with multi-input PE. 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 can not be decrypted. WE has several disrupting applications such as encrypting messages that can be decrypted in the 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. [BJK+18], an n-input PE for arbitrary predicates, secure in the secret key setting and without collusion, implies WE for NP and n-size witnesses. The construction is reminiscent to the one of iO from secret-key multi-input functional encryption [GGG+14]. Unfortunately, we cannot plug in our construction of n-input PE since it only supports conjunctions of arbitrary predicates (see Equation (1)). Currently, it is not know how to build multi-input PE for arbitrary predicates without iO.

It may seem that arbitrary predicates are a necessary condition in order to build WE from multi-input PE. However, we highlight that this is not necessarily the case if we consider security under corruptions. In particular, a 2-input PE for conjunctions under 1 corruption and no collusions, implies WE for any relation. This can be accomplished by considering the predicate $\mathbb{P}_{x,\mathcal{R}}(\cdot,\cdot) = \mathbb{P}_0(\cdot) \wedge \mathbb{P}_{x,\mathcal{R}}(\cdot)$ such that $\mathbb{P}_0(x_0^\star) = 1$ (for some wildcard $x_0^\star$) and $\mathbb{P}_{x,\mathcal{R}}(\omega) = (x, \omega) \in \mathcal{R}$. Intuitively, to encrypt m using a statement x, the sender can simply output $c_0, \mathsf{ek}_1, \mathsf{dk}_{\mathbb{P}_{\mathcal{R}}}$ such that $c_0 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_0, x_0^\star, m)$, $\mathsf{dk}_{\mathbb{P}_{x,\mathcal{R}}} \leftarrow\!\!\$\ \mathsf{KGen}(m, \mathbb{P}_{x,\mathcal{R}})$, and $(\mathsf{msk}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$. Then, the receiver uses w to retrieve m by computing $\mathsf{Dec}(\mathsf{dk}_{x,\mathcal{R}}, c_0, \mathsf{Enc}(\mathsf{ek}_1, w))$.[6] Here, security of the 2-input PE against corruptions is fundamental since it allows the sender to give ek_1 to the (possibly malicious) receiver and give him the opportunity to try different witnesses.

Unfortunately, even in this case, our construction of $O(1)$-input PE under corruptions fails to imply WE. This is because our construction supports conjunctions of arbitrary predicates *each one* having a wildcard. In other words, we do not support predicate that are unsatisfiable. This is a problem because supporting unsatisfiable predicates is crucial to prove security of WE when $x \notin \mathcal{L}$.[7] Given this technical hurdle, we identify two plausible approaches that would lead to a construction of WE:

- Enlarging the class of predicates of our secret-key n-input PE construction: From conjunction of arbitrary predicates with wildcards (see Equation (1)) to arbitrary n-input

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

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

predicates.

- Supporting conjunctions of arbitrary predicates (without wildcards) in the setting of 2-input PE with security under 1 corruption.

3 Preliminaries

3.1 Notation

We use the notation $[n] = \{1, 2, \ldots, n\}$ and $[a, b] = \{a, a+1, \ldots, b\}$ for $a > b$ (resp. $[a, b] = \{a, a-1, \ldots, b\}$ for $a < b$). 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 $x \in \{0,1\}^*$, 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 $x, r \in \{0,1\}^*$ 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.

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 [GKW17, WZ17] 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:

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

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:

$\mathsf{Obf}(1^\lambda, \mathbb{C}, y, m)$**:** Upon input the security parameter 1^λ, a circuit $\mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$, a lock $y \in \{0,1\}^{s(\lambda)}$, 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 obuscated circuit $\widetilde{\mathbb{C}}$ and an input $x \in \{0,1\}^{n(\lambda)}$, the deterministic evaluation algorithm outputs a message $m \in \mathcal{M} \cup \{\bot\}$.

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

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

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

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

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

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

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

Else **return** 0

Figure 1: Game defining security of lockable obfuscation.

1. $\forall \lambda \in \mathbb{N}$, $\forall x \in \{0,1\}^{n(\lambda)}$, $m \in \mathcal{M}$, $\forall \mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$ such that $\mathbb{C}(x) = y$, we have

$$\mathbb{P}\Big[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, \mathbb{C}, y, m), x) = m\Big] = 1.$$

2. $\forall \lambda \in \mathbb{N}$, $\forall x \in \{0,1\}^{n(\lambda)}$, $\forall m \in \mathcal{M}$, $\forall \mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$ such that $\mathbb{C}(x) \neq y$, we have

$$\mathbb{P}\Big[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, \mathbb{C}, y, m), x) = \perp\Big] \geq 1 - \mathsf{negl}(\lambda).$$

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:

$$\left|\mathbb{P}\Big[\mathbf{G}^{\mathsf{lock\text{-}sim}}_{\Pi,\mathsf{A},\mathsf{S}}(\lambda) = 1\Big] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda), \tag{3}$$

where $\mathbf{G}^{\mathsf{lock\text{-}sim}}_{\Pi,\mathsf{A},\mathsf{S}}(\lambda)$ is depicted in Figure 1.

Remark 1. The definitions above are taken from [GKW17]. Wichs and Zirdelis [WZ17] 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.

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.

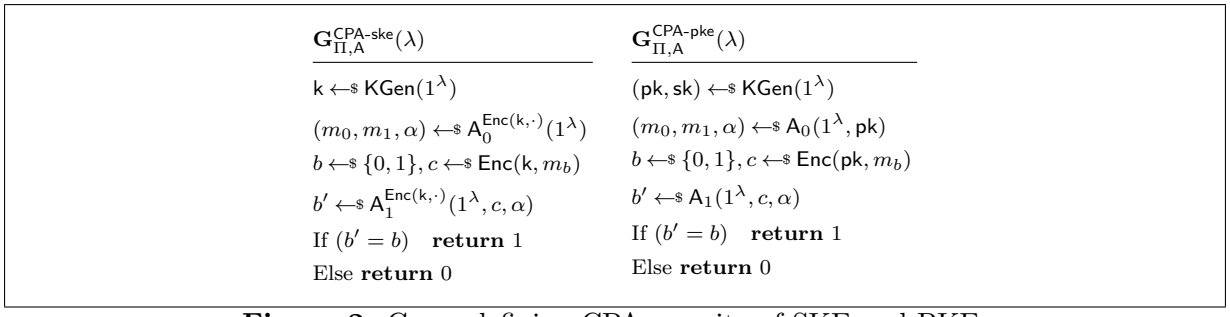

Figure 2: Game defining CPA security of SKE and PKE.

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 $\forall\lambda \in \mathbb{N}$, $\forall\mathsf{k}$ output by $\mathsf{KGen}(1^\lambda)$, $\forall m \in \mathcal{M}$, we have

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

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 Figure 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

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

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 Figure 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 $\mathbb{P}$ (corresponding to its decryption key) is satisfied when evaluated with the predicate input x used for encrypting the plaintext, i.e. $\mathbb{P}(x) = 1$. 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.

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

$\mathsf{Enc}(\mathsf{mpk}, x, m)$**:** 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.

$\mathsf{Dec}(\mathsf{dk}_\mathbb{P}, c)$**:** The deterministic decryption algorithm takes as input a secret decryption key $\mathsf{dk}_\mathbb{P}$ for predicate $\mathbb{P} \in \mathcal{P}$ 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 $\mathbb{P}(x) = 1$. On the other hand, if $\mathbb{P}(x) = 0$, 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 $\forall\lambda \in \mathbb{N}$, $\forall(\mathsf{mpk}, \mathsf{msk})$ output by $\mathsf{Setup}(1^\lambda)$, $\forall m \in \mathcal{M}$, $\forall x \in \mathcal{X}$,$\forall\mathbb{P} \in \mathcal{P}$, the following two conditions hold:

Decryptable ciphertexts: If $\mathbb{P}(x) = 1$, then

$$\mathbb{P}[\mathsf{Dec}(\mathsf{dk}_\mathbb{P}, \mathsf{Enc}(\mathsf{mpk}, x, m)) = m : \mathsf{dk}_\mathbb{P} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P})] \geq 1 - \mathsf{negl}(\lambda).$$

Non-decryptable ciphertexts: If $\mathbb{P}(x) = 0$, then

$$\mathbb{P}[\mathsf{Dec}(\mathsf{dk}_\mathbb{P}, \mathsf{Enc}(\mathsf{mpk}, x, m)) = \perp : \mathsf{dk}_\mathbb{P} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P})] \geq 1 - \mathsf{negl}(\lambda).$$

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.[8]

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

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

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

[8] We stress that PE schemes that only satisfy CPA security are also called attribute-based encryption (ABE) schemes.

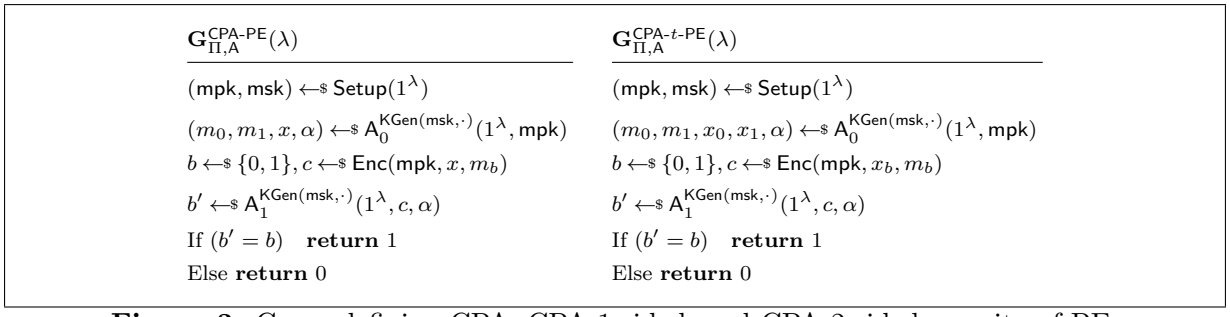

Figure 3: Game defining CPA, CPA-1-sided, and CPA-2-sided security of PE.

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 $t \in \{1,2\}$. We say that a PE Π is CPA-t-sided secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

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

Case $t = 1$**:** $\mathbb{P}(x_0) = \mathbb{P}(x_1) = 0$.
Case $t = 2$**:** Either $\mathbb{P}(x_0) = \mathbb{P}(x_1) = 0$ or $\mathbb{P}(x_0) = \mathbb{P}(x_1) \wedge m_0 = m_1$.

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 can not get more than one decryption key.

4 Multi-key and Multi-input Predicate Encryption

As already discussed in the introduction, a multi-key PE scheme is defined over an ensemble of predicates $\mathcal{P} = \{\mathbb{P}_v\}_{v\in\mathcal{V}}$ indexed by a value $v \in \mathcal{V}$ which is uniquely represented as a sequence $v = (v_0, \ldots, v_{n-1}) \in \mathcal{V}_0 \times \ldots \times \mathcal{V}_{n-1}$. A trusted authority generates decryption keys dk_{v_i} for each $i \in [0, n-1]$, with the guarantee that, given the decryption keys $\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}$, the receiver can decrypt successfully the ciphertext c (associated to plaintext m and input x), so long as $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x) = 1$.

In multi-input PE instead, we consider predicates $\mathbb{P}$ with n-arity, i.e., predicates of the form $\mathbb{P}(x_0, \ldots, x_{n-1})$. A trusted authority produces encryption keys ek_i which are associated to the i-th input for $\mathbb{P}$; namely, given ek_i, a sender can generate a ciphertext c_i which is an encryption of message m_i under input x_i. At the same time, the authority can produce a decryption key $\mathsf{dk}_{\mathbb{P}}$ (using a master secret key msk) associated to a predicate $\mathbb{P}$. As usual, the receiver can successfully decrypt $c_0, \ldots, c_{n-1}$ (and thus obtain $m_0, \ldots, m_{n-1}$), so long as $\mathbb{P}(x_0, \ldots, x_{n-1}) = 1$.

We provide the formal definitions of multi-key PE and multi-input PE in Section 4.1 and Section 4.2, respectively.

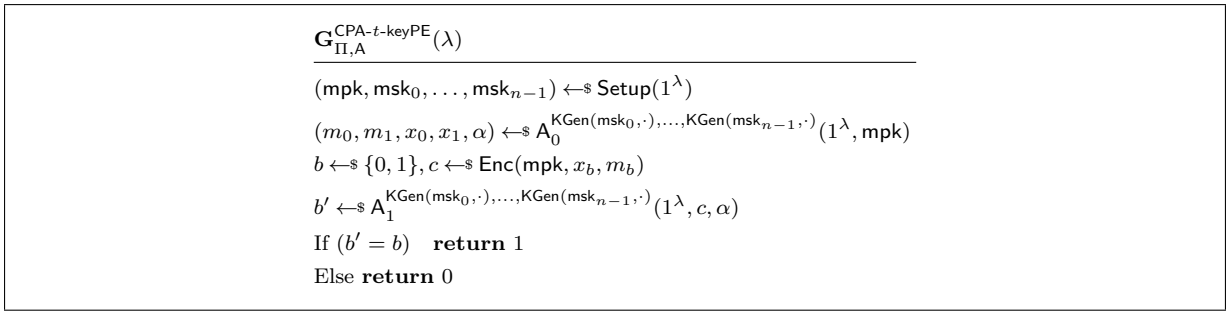

Figure 4: Game defining CPA-t-sided security of n-key PE.

4.1 Multi-key PE

Formally, an n-key PE with message space $\mathcal{M}$, input space $\mathcal{X}$, and predicate space $\mathcal{P} = \{\mathbb{P}_{v_0,\ldots,v_{n-1}}\}_{(v_0,\ldots,v_{n-1})\in\mathcal{V}}$ indexed by $\mathcal{V} = \mathcal{V}_0 \times \ldots \times \mathcal{V}_{n-1}$, 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 n master secret key $(\mathsf{msk}_0, \ldots, \mathsf{msk}_{n-1})$.

$\mathsf{KGen}(\mathsf{msk}_i, v_i)$**:** Let $i \in [0, n-1]$. The randomized key generator takes as input the i-th master secret key msk_i and the i-th index $v_i \in \mathcal{V}_i$. The algorithm outputs the i-th 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.

$\mathsf{Dec}(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}, c)$**:** The deterministic decryption algorithm takes as input n secret decryption keys $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}})$ for the n indexes $(v_0, \ldots, v_{n-1}) \in \mathcal{V}$ and a ciphertext c. The algorithm outputs either a message m, or an error $\perp$.

Correctness is intuitive: given the decryption keys $(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}})$ for $(v_0, \ldots, v_{n-1}) \in \mathcal{V}$, the decryption algorithm returns the message m (encrypted under the input x) with overwhelming probability, whenever $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x) = 1$.

Definition 10 (Correctness of n-key PE)**.** A n-key PE with message space $\mathcal{M}$, input space $\mathcal{X}$, predicate space $\mathcal{P} = \{\mathbb{P}_{v_0,\ldots,v_{n-1}}\}_{v_0,\ldots,v_{n-1}\in\mathcal{V}}$ indexed by $\mathcal{V} = \mathcal{V}_0 \times \ldots \times \mathcal{V}_{n-1}$, is correct if $\forall\lambda \in \mathbb{N}$, $\forall(\mathsf{mpk}, \mathsf{msk}_0, \ldots, \mathsf{msk}_{n-1})$ output by $\mathsf{Setup}(1^\lambda)$, $\forall m \in \mathcal{M}$, $\forall x \in \mathcal{X}$, $\forall(v_0, \ldots, v_{n-1}) \in \mathcal{V}$ such that $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x) = 1$, we have:

$$\mathbb{P}\big[\mathsf{Dec}(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}, \mathsf{Enc}(\mathsf{mpk}, x, m)) = m\big] \geq 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, v_i)$ for $i \in [0, n-1]$.

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 $i \in [0, n-1]$) 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 $(v_0, \ldots, v_{n-1})$ such that $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 0$. Analogously, the CPA-2-sided security captures the indistinguishability of $\mathsf{Enc}(\mathsf{mpk}, x_0, m_0)$ and $\mathsf{Enc}(\mathsf{mpk}, x_1, m_1)$ even when the adversary can decrypt the challenge ciphertext, i.e., $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 1$ 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 $t \in \{1,2\}$. We say that a n-key PE Π is CPA-t-sided secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{CPA}\text{-}t\text{-}\mathsf{keyPE}}(\lambda)$ is depicted in Figure 4. Adversary A is called valid if $\forall v_0 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_0,\cdot)}, \ldots, v_{n-1} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{n-1},\cdot)}$, we have

$$\begin{aligned} &\textbf{Case } t = 1\textbf{:}\ \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 0. \\ &\textbf{Case } t = 2\textbf{:}\ \text{Either } \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 0 \text{ or} \\ &\qquad\qquad \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) \wedge m_0 = m_1. \end{aligned}$$

4.2 Multi-input PE

Formally, an n-input PE with message space $\mathcal{M} = \mathcal{M}_0 \times \ldots \times \mathcal{M}_{n-1}$, input space $\mathcal{X} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_{n-1}$, 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 encryption keys $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1})$ and the master secret key msk.

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

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$**:** Let $i \in [0, n-1]$. The randomized encryption algorithm takes as input an encryption key ek_i, an input $x \in \mathcal{X}_i$, and a message $m \in \mathcal{M}_i$. The algorithm produces a ciphertext c linked to x_i.

$\mathsf{Dec}(\mathsf{dk}_\mathbb{P}, c_0, \ldots, c_{n-1})$**:** The deterministic decryption algorithm takes as input a secret decryption key $\mathsf{dk}_\mathbb{P}$ for predicate $\mathbb{P} \in \mathcal{P}$ and n ciphertexts $(c_0, \ldots, c_{n-1})$. The algorithm outputs either n messages $(m_0, \ldots, m_{n-1})$, or an error $\perp$.

Correctness states that ciphertexts $(c_0, \ldots, c_{n-1})$, each linked to an input x_i, correctly decrypt with overwhelming probability if $\mathbb{P}(x_0, \ldots, x_{n-1}) = 1$.

Definition 12 (Correctness of n-input PE)**.** A n-input PE with message space $\mathcal{M} = \mathcal{M}_0 \times \ldots \times \mathcal{M}_{n-1}$, input space $\mathcal{X} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_{n-1}$, predicate space $\mathcal{P}$, is correct if $\forall \lambda \in \mathbb{N}$, $\forall(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk})$ output by $\mathsf{Setup}(1^\lambda)$, $\forall(m_0, \ldots, m_{n-1}) \in \mathcal{M}$, $\forall(x_0, \ldots, x_{n-1}) \in \mathcal{X}$,$\forall \mathbb{P} \in \mathcal{P}$ such that $\mathbb{P}(x_0, \ldots, x_{n-1}) = 1$, we have:

$$\mathbb{P}[\mathsf{Dec}(\mathsf{dk}_\mathbb{P}, c_0, \ldots, c_{n-1}) = (m_0, \ldots, m_{n-1})] \geq 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{dk}_\mathbb{P} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P})$ and $c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$ for $i \in [0, n-1]$.

4.2.1 Security with and without corruptions

The CPA-1-sided and CPA-2-sided security of n-input PE capture the infeasibility in distinguishing between $(\mathsf{Enc}(\mathsf{ek}_0, x_0^0, m_0^0), \ldots, \mathsf{Enc}(\mathsf{ek}_{n-1}, x_{n-1}^0, m_{n-1}^0))$ and $(\mathsf{Enc}(\mathsf{ek}_0, x_0^1, m_0^1), \ldots, \mathsf{Enc}(\mathsf{ek}_{n-1}, x_{n-1}^1, m_{n-1}^1))$. This is modeled by an adversary having oracle access to a key generation oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$ (allowing it to get decryption keys $\mathsf{dk}_\mathbb{P}$ on predicates of its choice) and n encryption oracles $\mathsf{Enc}(\mathsf{ek}_0, \cdot, \cdot), \ldots, \mathsf{Enc}(\mathsf{ek}_{n-1}, \cdot, \cdot)$ (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

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

$(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$

$(((m^i_0, \ldots, m^i_{n-1}), (x^i_0, \ldots, x^i_{n-1}))_{i\in\{0,1\}}, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Corr}(\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot),\}_{j\in[0,n-1]}}(1^\lambda)$

$b \leftarrow\!\!\$\ \{0,1\}, c_0 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_0, x^b_0, m^b_0), \ldots, c_{n-1} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_{n-1}, x^b_{n-1}, m^b_{n-1})$

$b' \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Corr}(\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[0,n-1]}}(1^\lambda, c_0, \ldots, c_{n-1}, \alpha)$

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

Else **return** 0

Figure 5: Game defining CPA-t-sided security of n-input PE in the ℓ-corruption setting. Oracle $\mathsf{Corr}(j)$ returns ek_j for $j \in [0, n-1]$.

(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 $i \in [0, n-1]$, 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 i-th ciphertexts on arbitrary i-th inputs x_i and potentially decrypt part of the challenge ciphertext. Concretely, as for CPA-1-sided security, the validity of the adversary can be defined as follows:

- *No corruptions (a.k.a. the secret-key setting).* If all the encryption keys $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1})$ are secret the validity conditions of CPA-1-sided security is straightforward. It intuitively states that for every $\mathsf{dk}_\mathbb{P}$ (obtained through oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$) and any tuple of ciphertexts $(c_0, \ldots, c_{n-1})$ (each linked to an input x_i) obtained through the interleaving of part of the challenge ciphertext with the ciphertexts generated by invoking oracles $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i\in[0,n-1]}$, we must have that $\mathbb{P}(x_0, \ldots, x_{n-1}) = 0$ (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 i-th encryption key ek_i. Taking into account this observation, the validity of CPA-1-sided security *with corruptions* says that any tuple of ciphertexts $(c_0, \ldots, c_{n-1})$ that can be obtained by interleaving part of the challenge ciphertexts with both the ones generated through oracles $\{\mathsf{Enc}(\mathsf{ek}_i, \cdot, \cdot)\}_{i\in[0,n-1]}$ and the ones that can be autonomously generated using the leaked encryption keys, we must have that $\mathbb{P}(x_0, \ldots, x_{n-1}) = 0$.

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

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

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

where game $\mathbf{G}^{\ell\text{-CPA-}t\text{-inputPE}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Figure 5. Let $\mathcal{Q}_i = \{x | \exists (x, m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_i,\cdot,\cdot)}\}$ for $i \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}(\cdot)}$ and $\mathcal{Q}_i = \mathcal{X}_i$ for $i \in \mathcal{Q}_{\mathsf{Corr}(\cdot)}$. Adversary A is called valid if $|\mathcal{Q}_{\mathsf{Corr}}| \leq \ell$ and

$\forall \mathbb{P} \in \mathcal{Q}_{\mathsf{KGen}}$, $\forall d \in \{0,1\}$, $\forall (x'_0, \ldots, x'_{n-1}) \in \mathcal{Q}_0 \cup \{x_0^d\} \times \ldots \times \mathcal{Q}_{n-1} \cup \{x_{n-1}^d\}$, $\forall j \in \{0, \ldots, n-1\}$, we have

$$\textbf{Case } t = 1\textbf{:}\ \mathbb{P}(x'_0, \ldots, x'_{j-1}, x_j^d, x'_{j+1}, \ldots, x'_{n-1}) = 0 \tag{4}$$

$$\begin{aligned}\textbf{Case } t = 2\textbf{:}\ &\text{Either } \mathbb{P}(x'_0, \ldots, x'_{j-1}, x_j^d, x'_{j+1}, \ldots, x'_{n-1}) = 0\\ &\text{or } \mathbb{P}(x'_0, \ldots, x'_{j-1}, x_j^d, x'_{j+1}, \ldots, x'_{n-1}) =\\ &\quad \mathbb{P}(x'_0, \ldots, x'_{j-1}, x_j^{1-d}, x'_{j+1}, \ldots, x'_{n-1}) \wedge\\ &\quad m_0^d = m_0^{1-d} \wedge \ldots \wedge m_{n-1}^d = m_{n-1}^{1-d}.\end{aligned} \tag{5}$$

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

4.2.2 Relation with multi-key PE

We notice that $(n+1)$-input PE, tolerating 1 corruption, naturally implies n-key PE. The idea is to use the first n inputs of the predicate $\mathbb{P}(x_0, \ldots, x_n)$ (of $(n+1)$-input PE) to determine the indexes $(v_0, \ldots, v_n) \in \mathcal{V}$ that define the predicate $\mathbb{P}_{x_0,\ldots,x_{n-1}}(x_n)$ of the n-key PE, i.e.,

$$\mathbb{P}(x_0, \ldots, x_n) = \mathbb{P}(v_0, \ldots, v_{n-1}, x) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x),$$

where $x_i = v_i$ for $i \in [0, n-1]$ and $x_n = x$. In a nutshell, the authority uses the Setup algorithm of the $(n+1)$-input PE to generate $n+1$ encryption keys $(\mathsf{ek}_0, \ldots, \mathsf{ek}_n)$ together with the master secret key msk. Then, it uses the latter to compute $\mathsf{dk}_{\mathbb{P}} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P})$ and set $\mathsf{mpk} = \mathsf{ek}_n$ and $\mathsf{msk}_i = (\mathsf{dk}_{\mathbb{P}}, \mathsf{ek}_i)$ for $i \in [0, n-1]$. To generate dk_{v_i} (for an index v_i) the authority computes $c_{v_i} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_i, v_i, \bot)$ (using the encryption algorithm of $(n+1)$-input PE) and outputs $\mathsf{dk}_{v_i} = (\mathsf{dk}_{\mathbb{P}}, c_{v_i})$. At this point, the n-key PE encryption and decryption algorithms are straightforward. To encrypt a message m under an input x, the sender computes $c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_n, x, m)$. On the other hand, the receiver decrypts the message by executing $\mathsf{Dec}(\mathsf{ek}_{\mathbb{P}}, c_{v_0}, \ldots, c_{v_{n-1}}, c)$ where $\mathsf{mpk} = \mathsf{ek}_n$ and $\mathsf{dk}_{v_i} = (\mathsf{dk}_{\mathbb{P}}, c_{v_i})$ for $i \in [0, n-1]$. Intuitively, the CPA-t-sided security of this n-key PE follows from the CPA-t-sided security in the 1-corruption setting of the $(n+1)$-input PE. Observe that the $(n+1)$-input PE must tolerate 1 corruption since ek_n (i.e., the mpk of the n-key PE) is made public.

For completeness, we highlight that the same construction works if we use an $(n+1)$-input PE that satisfies a much weaker flavor of security under corruptions, named the *ℓ-hybrid setting*. Here, the Setup algorithm takes an additional parameter 1^ℓ that determines the number of encryption keys that we want to make public.[9] Hence, in this setting the adversary does not have access to the corruption oracle $\mathsf{Corr}(\cdot)$ but, instead, it directly receives as input the public encryption keys generated on setup. Clearly, security in the ℓ-hybrid setting is stronger than security in the secret-key setting but weaker than security in the $(\ell > 0)$-corruption setting). We provide the formal definition of security in the ℓ-hybrid setting and show that it suffices to imply multi-key PE in Appendix B.

[9] In other words, this is equivalent to saying that the corruptions are known ahead of time and the construction can depend on this information.

5 Constructions

In this section, we give different constructions of multi-key and multi-input PE (see also Section 2) for the following class of predicates:

$$\mathbb{P}(x_0, \ldots, x_{n-1}) = \mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1}). \tag{6}$$

In particular, Section 5.1 we give a construction of n-key PE from single-input PE and lockable obfuscation for $n \in \mathsf{poly}(\lambda)$. This construction is secure against unbounded collusion. In Section 5.2, 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 secure without collusion and in the secret key setting. The second handles $O(1)$-airity and it is CPA-1-side secure without collusion and in the $(n-1)$-corruption setting. This second construction leverages a new nesting execution technique of (lockable obfuscated) circuits. Both multi-input construction support conjunctions of arbitrary predicates (Equation (6)) with wildcards, i.e., for every predicate $\mathbb{P}_i$ there exists $x_i^\star$ such that $\mathbb{P}_i(x_i^\star) = 1$. Hence, the multi-input constructions do not support unsatisfiable predicates.

5.1 Multi-key PE from PE and Lockable Obfuscation

Construction 1. *Consider the following primitives:*

1. *For $i \in [0, n-1]$, a PE scheme $\mathsf{PE}_i = (\mathsf{Setup}_i, \mathsf{KGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ with message space $\mathcal{M}_i$, input space $\mathcal{X}_i$, and predicate space $\mathcal{P}_i = \{\mathbb{P}_v\}_{v \in \mathcal{V}_i}$ indexed by $\mathcal{V}_i$. Without loss of generality, we assume that PE_i has ciphertext space $\mathcal{Y}_i$, $\mathcal{M}_0 = \{0,1\}^{m(\lambda)}$, and $\mathcal{M}_i = \mathcal{Y}_{i-1}$ for every $i \in [1, n-1]$.*

2. *A lockable obfuscation scheme $\mathsf{LOBF} = (\mathsf{Obf}, \mathsf{Eval})$ with message space $\mathcal{M}$ for the family of circuits $\mathcal{C}_{n,s,d}(\lambda) = \{\mathbb{C}_c\}$ defined as follows:*

   $\mathbb{C}_c(\mathsf{dk}_0, \ldots, \mathsf{dk}_{n-1})$
   Initialize $c_{n-1} = c$.
   For $i \in [n-1, 0]$ do:
   $\quad \mathsf{Dec}_i(\mathsf{dk}_i, c_i) = c_{i-1}$.
   end for.
   return c_{-1}.

   for $n(\lambda)$, $s(\lambda)$, $d(\lambda)$ that depends on the PE schemes $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$ used, and the circuit depth of the circuits $\mathcal{C}_{n,s,d}(\lambda)$.

We build a n-key PE scheme Π with message space $\mathcal{M}$, input space $\mathcal{X} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_{n-1}$, and predicate space $\mathcal{P} = \{\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0, \ldots, x_{n-1}) = \mathbb{P}_{v_0}(x_0) \wedge \ldots \wedge \mathbb{P}_{v_{n-1}}(x_{n-1})\}_{(v_0,\ldots,v_{n-1}) \in \mathcal{V}}$ indexed by $\mathcal{V} = \mathcal{V}_0 \times \ldots \times \mathcal{V}_{n-1}$ (and $\mathbb{P}_{v_i} \in \mathcal{P}_i$ for $i \in [0, n-1]$), in the following way:

$\mathsf{Setup}(1^\lambda)$**:** *Upon input the security parameter 1^λ the randomized setup algorithm outputs $\mathsf{mpk} = (\mathsf{mpk}_0, \ldots, \mathsf{mpk}_{n-1})$ and $\mathsf{msk}_0, \ldots, \mathsf{msk}_{n-1}$ where $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow\!\!\$\ \mathsf{Setup}_i(1^\lambda)$ for $i \in [0, n-1]$.*

$\mathsf{KGen}(\mathsf{msk}_i, v_i)$**:** *Let $i \in [0, n-1]$. Upon input the i-th master secret key msk_i and the i-th predicate's index $v_i \in \mathcal{V}_i$, the randomized key generator outputs $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_1, \mathbb{P}_{v_i})$ where $\mathbb{P}_{v_i} \in \mathcal{P}_i$.*

$\mathsf{Enc}(\mathsf{mpk}, x, m)$: *Upon input the master public key* $\mathsf{mpk} = (\mathsf{mpk}_0, \ldots, \mathsf{mpk}_{n-1})$, *an input* $x = (x_0, \ldots, x_{n-1}) \in \mathcal{X}$, *and a message* $m \in \mathcal{M}$, *the randomized encryption proceeds as follows:*

1. *Sample* $y \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ *and let* $c_{-1} = y$.
2. *For* $i \in [0, n-1]$, *compute* $c_i \leftarrow\!\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x_i, c_{i-1})$.

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

$\mathsf{Dec}(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}, c)$: *Upon input* n *secret decryption key* $\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}$ *and a ciphertext* $c = \widetilde{\mathbb{C}}$, *the deterministic decryption algorithm outputs* $m = \mathsf{Eval}(\widetilde{\mathbb{C}}, (\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}))$.

Correctness follows from the correctness of the underlying schemes. Below, we establish the following result whose proof appears in Appendix C.1

Theorem 4. *Let* $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$ *and* LOBF *be as above.*

1. *If each* $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$ *is CPA secure (Definition 8) and* LOBF *is secure (Definition 2), then the* n*-key PE scheme* Π *from Construction 1 is CPA-1-sided secure (Definition 11).*
2. *If each* $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$ *is CPA-2-sided secure (Definition 9) and* LOBF *is secure (Definition 2), then the* n*-key PE scheme* Π *from Construction 1 is CPA-2-sided secure (Definition 11).*

5.2 Multi-input PE from PE, Lockable Obfuscation and SKE/PKE

5.2.1 Secret key setting

First, we present our n-input PE construction that is CPA-1-sided secure in the secret key setting without collusion. It leverages single-input PE, lockable obfuscation, and SKE.

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

1. *A PE scheme* $\mathsf{PE} = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *with message space* $\mathcal{M}_1 = \{0,1\}^{m(\lambda)} \times \mathcal{M}_1'$, *input space* $\mathcal{X}_1 = \mathcal{X}_{1,0} \times \ldots \times \mathcal{X}_{1,n-1}$, *and predicate space* $\mathcal{P}_1 = \{\mathbb{P}(x_0, \ldots, x_{n-1})\} = \{\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})\}$. *Without loss of generality, we assume that* PE *has ciphertext space* $\mathcal{M}_2$ *and there exists a wildcard input* $(x_0^\star, \ldots, x_{n-1}^\star) \in \mathcal{X}_1$ *such that*

$$\forall (\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})) \in \mathcal{P}_1, \forall i \in [0, n-1], \mathbb{P}_i(x_i^\star) = 1.$$

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* $\mathcal{C}_{n,s,d}(\lambda) = \{\mathbb{C}_{c,\mathsf{k}}\}$ *defined as follows:*

$\mathbb{C}_{c,\mathsf{k}}(c_0, \ldots, c_{n-2}, \mathsf{dk}_{\mathbb{P}})$
Initialize $\mathsf{k}_0 = \mathsf{k}$ *and* $c_{n-1} = c$.
For $i \in [0, n-1]$ *do:*
$\mathsf{Dec}_1(\mathsf{dk}_{\mathbb{P}}, \mathsf{Dec}_2(\mathsf{k}_i, c_i)) = v_i$.
If $v_i = \perp$, **return** $\perp$.
Otherwise, let $v_i = (y_i \| \mathsf{k}_{i+1})$.
end for.
return y_{n-1}.

for $n(\lambda)$, $s(\lambda)$, $d(\lambda)$ that depends on PE scheme PE *used, and the circuit depth of the circuits $\mathcal{C}_{n,s,d}(\lambda)$.*

We build a n-input PE scheme with message space $\mathcal{M} = \mathcal{M}_{3,0} \times \ldots \times \mathcal{M}_{3,n-1}$ such that $\mathcal{M}_{3,i} = \mathcal{M}_3$ for $i \in [0, n-1]$, input space $\mathcal{X}_1 = \mathcal{X}_{1,0} \times \ldots \times \mathcal{X}_{1,n-1}$, and predicate space $\mathcal{P}_1 = \{\mathbb{P}(x_0, \ldots, x_{n-1})\} = \{\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})\}$ with wildcards (i.e., $\forall \mathbb{P} \in \mathcal{P}_1$, $\exists x_i^\star \in \mathcal{X}_{1,i}$ such that $\mathbb{P}_i(x_i^\star) = 1$), in the following way:

$\mathsf{Setup}(1^\lambda)$**:** *Upon input the security parameter 1^λ, the randomized setup algorithm outputs $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1})$ and msk where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$, $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{k}_i, \mathsf{k}_{i+1})$, $\mathsf{k}_n = \mathsf{k}_0$, and $\mathsf{k}_i \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ for $i \in [0, n-1]$.*

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

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$**:** *Let $i \in [0, n-1]$. 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 $y_i \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and compute $c_i^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_i || \mathsf{k}_{i+1})$ where $x_j = x_j^\star$ for any $j \in [0, n-1] \backslash \{i\}$. 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 $c_i^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_i, c_i^{(1)})$.*

$\mathsf{Dec}(\mathsf{dk}_\mathbb{P}, c_0, \ldots, c_{n-1})$**:** *Upon input a secret decryption key $\mathsf{dk}_\mathbb{P}$ for predicate $\mathbb{P} \in \mathcal{P}$, and n ciphertexts $(c_0, \ldots, c_{n-1})$ such that $c_i = (\widetilde{\mathbb{C}}_i, c_i^{(2)})$ for $i \in [0, n-1]$. The deterministic decryption returns $(m_0, \ldots, m_{n-1})$ where $m_i = \mathsf{Eval}(\widetilde{\mathbb{C}}_i, (c_{i+1}^{(2)}, \ldots, c_{n-1}^{(2)}, c_0^{(2)}, \ldots, c_{i-1}^{(2)}, \mathsf{dk}_\mathbb{P}))$ for $i \in [0, n-1]$.*

As usual, correctness follows from the correctness of the underlying primitives. Below, we establish the following result whose proof appear in Appendix C.2.

Theorem 5. *Let* PE, SKE, *and* LOBF *be as above. If* PE *is CPA-1-sided secure without collusion (Definition 8),* SKE *is CPA secure (Definition 4), and* LOBF *is secure (Definition 2), then the n-input PE scheme Π from Construction 2 is CPA-1-sided secure in the secret key setting without collusion (Definition 13).*

5.2.2 Corruption setting

We now move on our construction of n-input PE secure that is CPA-1-sided secure in the $(n-1)$-corruption setting without collusion. This construction handles constant-arity (i.e., $n \in O(1)$) and it is based on single-input PE, lockable obfuscation, and PKE. It leverages the nested execution technique described in Section 2.

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

1. *A PE scheme $\mathsf{PE} = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ with message space $\mathcal{M}_1 = \{0,1\}^{m_3(\lambda)+m_4(\lambda)}$, input space $\mathcal{X}_1 = \mathcal{X}_{1,0} \times \ldots \times \mathcal{X}_{1,n-1}$, and predicate space $\mathcal{P}_1 = \{\mathbb{P}(x_0, \ldots, x_{n-1})\} = \{\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})\}$. Without loss of generality, we assume that PE has ciphertext space $\mathcal{Y}_1$ and there exists a wildcard input $(x_0^\star, \ldots, x_{n-1}^\star) \in \mathcal{X}_1$ such that*

$$\forall (\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_n(x_{n-1})) \in \mathcal{P}_1, \forall i \in [0, n-1], \mathbb{P}_i(x_i^\star) = 1.$$

2. *For* $i \in [0, n-1]$, *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* $\mathcal{M}_{2,0} = \mathcal{Y}_1$, *and* $\mathcal{M}_{2,i} = \mathcal{Y}_{2,i-1}$ *for every* $i \in [1, n-1]$

3. *A lockable obfuscation scheme* $\mathsf{LOBF}_3 = (\mathsf{Obf}_3, \mathsf{Eval}_3)$ *with message space* $\mathcal{M}_3 = (\mathcal{K}_{2,0} \cup \ldots \cup \mathcal{K}_{2,n-1}) \times \{0,1\}^{\lfloor \log_2(n-1) \rfloor + 1}$ *for the family of circuits* $\mathcal{C}^{\mathsf{in}}_{n_3,s_3,d_3}(\lambda) = \{\mathbb{C}^{\mathsf{in}}_{c,\mathsf{sk},i}\}$ *depicted in Figure 6, where* $n_3(\lambda)$, $s_3(\lambda)$, $d_3(\lambda)$ *that depends on PE and PKE schemes* $(\mathsf{PE}, \mathsf{PKE}_{2,0}, \ldots, \mathsf{PKE}_{2,n-1})$ *used, and the depth of 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* $\mathcal{C}^{\mathsf{out}}_{n_4,s_4,d_4}(\lambda) = \{\mathbb{C}^{\mathsf{out}}_{c,\mathsf{sk},i}\}$ *depicted in Figure 6, where* $n_4(\lambda)$, $s_4(\lambda)$, $d_4(\lambda)$ *that depends on PE and PKE schemes* $(\mathsf{PE}, \mathsf{PKE}_{2,0}, \ldots, \mathsf{PKE}_{2,n-1})$ *used, and the depth of the circuits* $\mathcal{C}^{\mathsf{out}}_{n_4,s_4,d_4}(\lambda)$.

We build a n-input PE scheme with message space $\overbrace{\mathcal{M}_4 \times \ldots \times \mathcal{M}_4}^{n}$, *input space* $\mathcal{X}_1 = \mathcal{X}_{1,0} \times \ldots \times \mathcal{X}_{1,n-1}$, *and predicate space* $\mathcal{P}_1 = \{\mathbb{P}(x_0, \ldots, x_{n-1})\} = \{\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})\}$ *with wildcards (i.e.,* $\forall \mathbb{P} \in \mathcal{P}_1$, $\exists x^\star_i \in \mathcal{X}_{1,i}$ *such that* $\mathbb{P}_i(x^\star_i) = 1$*), in the following way:*

$\mathsf{Setup}(1^\lambda)$**:** *Upon input the security parameter* 1^λ *the randomized setup algorithm outputs* $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1})$ *and* msk *where* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$, $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{sk}_i, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$, *and* $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\!\$\ \mathsf{KGen}_{2,i}(1^\lambda)$ *for* $i \in [0, n-1]$.

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

$\mathsf{Enc}(\mathsf{ek}_i, x_i, m_i)$**:** *Let* $i \in [0, n-1]$. *Upon input an encryption key* $\mathsf{ek}_i = (\mathsf{mpk}, \mathsf{sk}_i, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$, *an input* $x_i \in \mathcal{X}_{1,i}$, *and a message* $m_i \in \mathcal{M}_4$, *the randomized encryption algorithm samples* $(y^{\mathsf{in}}_i, y^{\mathsf{out}}_i) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$ *and prooceds as follows:*

1. *Compute* $c^{(-1)}_i \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}, y^{\mathsf{in}}_i || y^{\mathsf{out}}_i)$ *where* $x_j = x^\star_j$ *for* $j \in [0, n-1] \setminus \{i\}$.
2. *For* $j \in [0, n-1]$, *compute* $c^{(j)}_i \leftarrow\!\!\$\ \mathsf{Enc}_{2,j}(\mathsf{pk}_j, c^{(j-1)}_i)$.

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

$\mathsf{Dec}(\mathsf{dk}_{\mathbb{P}}, c_0, \ldots, c_{n-1})$**:** *Upon input a decryption key* $\mathsf{dk}_{\mathbb{P}}$ *for predicate* $\mathbb{P} \in \mathcal{P}_1$, *and n ciphertexts* $(c_0, \ldots, c_{n-1})$ *such that* $c_i = (\widetilde{\mathbb{C}}^{\mathsf{out}}_i, \widetilde{\mathbb{C}}^{\mathsf{in}}_i)$ *for* $i \in [0, n-1]$. *The deterministic decryption algorithm returns* $(m_0, \ldots, m_{n-1})$ *where* $m_i = \mathsf{Eval}_4(\widetilde{\mathbb{C}}^{\mathsf{out}}_i, (\widetilde{\mathbb{C}}^{\mathsf{in}}_0, \ldots, \widetilde{\mathbb{C}}^{\mathsf{in}}_{i-1}, \widetilde{\mathbb{C}}^{\mathsf{in}}_{i+1}, \ldots, \widetilde{\mathbb{C}}^{\mathsf{in}}_{n-1}, \mathsf{dk}_{\mathbb{P}}))$ *for* $i \in [0, n-1]$.

Correctness follows from the correctness of the underlying primitives. Below, we establish the following result whose proof appear in Appendix C.3.

Theorem 6. *Let* PE_1, $\mathsf{PKE}_{2,0}, \ldots, \mathsf{PKE}_{2,n-1}$, LOBF_3, *and* LOBF_4 *be as above. If* PE *is CPA secure without collusion (Definition 8), each* $\mathsf{PKE}_{2,i}$ *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 3 is CPA-1-sided secure in the* $(n-1)$*-corruptions setting without collusion (Definition 13).*

$\mathbb{C}^{\mathsf{in}}_{c,\mathsf{sk},i}(\mathbb{C}_0,\ldots,\mathbb{C}_{n-3},\mathsf{sk}_0,\ldots,\mathsf{sk}_{n-1},\mathsf{dk}_{\mathbb{P}})$

Initialize:

$c_{n-1}=c,\mathsf{sk}'_i=\mathsf{sk},\mathbb{C}_{n-2}=\bot.$

$\forall j\in[0,n-1]\setminus\{i\},\ \mathsf{sk}'_j=\mathsf{sk}_j.$

$k\in[0,n-3]$ such that $\mathbb{C}_k\neq\bot$ and $\mathbb{C}_{k+1}=\bot.$

end initialize.

If $\forall j\in[0,n-1],\mathsf{sk}'_j\neq\bot$:

For $j\in[n-1,0]$ do: $\mathsf{Dec}_{2,j}(\mathsf{sk}'_j,c_j)=c_{j-1}.$

If $\mathsf{Dec}_1(\mathsf{dk}_{\mathbb{P}},c_{-1})=v=\bot,$ **return** $\bot.$

Otherwise let $v=(y_i^{\mathsf{in}}||y_i^{\mathsf{out}}),$ **return** $y_i^{\mathsf{in}}.$

end if.

For $t\in[0,k]$ do:

$\mathsf{Eval}_3(\mathbb{C}_t,(\mathbb{C}_{t+1},\ldots,\mathbb{C}_k,\overbrace{\bot,\ldots,\bot}^{n-4+t-k},\mathsf{sk}'_0,\ldots,\mathsf{sk}'_{n-1},\mathsf{dk}_{\mathbb{P}}))=(\overline{\mathsf{sk}},h).$

If $\overline{\mathsf{sk}}=\bot,$ **return** $\bot.$

Otherwise let $\mathsf{sk}'_h=\overline{\mathsf{sk}}.$

end for.

For $j\in[n-1,0]$ do: $\mathsf{Dec}_{2,j}(\mathsf{sk}'_j,c_j)=c_{j-1}.$

If $\mathsf{Dec}_1(\mathsf{dk}_{\mathbb{P}},c_{-1})=v=\bot,$ **return** $\bot.$

Otherwise let $v=(y_i^{\mathsf{in}}||y_i^{\mathsf{out}}),$ **return** $y_i^{\mathsf{in}}.$

$\mathbb{C}^{\mathsf{out}}_{c,\mathsf{sk},i}(\mathbb{C}_0,\ldots,\mathbb{C}_{n-2},\mathsf{dk}_{\mathbb{P}})$

Initialize:

$c_{n-1}=c,\mathsf{sk}'_i=\mathsf{sk}.$

$\forall j\in[0,n-1]\setminus\{i\},\ \mathsf{sk}'_j=\bot.$

end initialize.

For $t\in[0,n-2]$ do:

$\mathsf{Eval}_3(\mathbb{C}_t,(\mathbb{C}_{t+1},\ldots,\mathbb{C}_{n-2},\overbrace{\bot,\ldots,\bot}^{t},\mathsf{sk}'_0,\ldots,\mathsf{sk}'_{n-1},\mathsf{dk}_{\mathbb{P}}))=(\overline{\mathsf{sk}},h).$

If $\overline{\mathsf{sk}}=\bot,$ **return** $\bot.$

Otherwise let $\mathsf{sk}'_h=\overline{\mathsf{sk}}.$

end for.

For $j\in[n-1,0]$ do: $\mathsf{Dec}_{2,j}(\mathsf{sk}'_j,c_j)=c_{j-1}.$

If $\mathsf{Dec}_1(\mathsf{dk}_{\mathbb{P}},c_{-1})=v=\bot,$ **return** $\bot.$

Otherwise let $v=(y_i^{\mathsf{in}}||y_i^{\mathsf{out}}),$ **return** $y_i^{\mathsf{out}}.$

Figure 6: Definitions of 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 3.

References

[ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho

Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 552–582. Springer, Heidelberg, December 2019.

[ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 128–157. Springer, Heidelberg, April 2019.

[ACF+18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Heidelberg, August 2018.

[AFNV19] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 701–731. Springer, Heidelberg, August 2019.

[AFNV21a] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: matchmaking encryption and its applications. *Journal of Cryptology*, 34(3):1–50, 2021.

[AFNV21b] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. *Journal of Cryptology*, 34(3):16, July 2021.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2011.

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, April / May 2017.

[AGT21] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 208–238, Virtual Event, August 2021. Springer, Heidelberg.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Heidelberg, May 2014.

[BDGM22] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. ICALP, 2022.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BGI+14] Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 387–404. Springer, Heidelberg, August 2014.

[BJK+18] Zvika Brakerski, Aayush Jain, Ilan Komargodski, Alain Passelègue, and Daniel Wichs. Non-trivial witness encryption and null-iO from standard assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 425–441. Springer, Heidelberg, September 2018.

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

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

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

[CDG+18] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, December 2018.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015.

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 245–277. Springer, Heidelberg, March 2018.

[FGRV21] Danilo Francati, Alessio Guidi, Luigi Russo, and Daniele Venturi. Identity-based matchmaking encryption without random oracles. In *International Conference on Cryptology in India*, pages 415–435. Springer, 2021.

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

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 736–749. ACM, 2021.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.

[HIJ+16] Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns. In Madhu Sudan, editor, *ITCS 2016*, pages 157–168. ACM, January 2016.

[HIJ+17] Shai Halevi, Yuval Ishai, Abhishek Jain, Ilan Komargodski, Amit Sahai, and Eylon Yogev. Non-interactive multiparty computation without correlated randomness. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 181–211. Springer, Heidelberg, December 2017.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 132–150. Springer, Heidelberg, August 2011.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Heidelberg, May / June 2022.

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

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

[LT19] Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 520–551. Springer, Heidelberg, December 2019.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.

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

[Tom19] Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 459–488. Springer, Heidelberg, December 2019.

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Heidelberg, August 2012.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Heidelberg, February 2014.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 127–156. Springer, Heidelberg, October 2021.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017.

A Matchmaking Encryption

A.1 Security 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 [AFNV19, AFNV21a], 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 [FGRV21, AFNV19, AFNV21b], 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.

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 14 (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 $\forall\lambda \in \mathbb{N}$, $\forall(\mathsf{mpk}, \mathsf{msk})$ output by $\mathsf{Setup}(1^\lambda)$, $\forall m \in \mathcal{M}$, $\forall\sigma \in \mathcal{U}_1$,$\forall\rho \in \mathcal{U}_2$, $\forall\mathbb{R} \in \mathcal{P}_1$, $\forall\mathbb{S} \in \mathcal{P}_2$ such that $\mathbb{S}(\sigma) = 1 \wedge \mathbb{R}(\rho) = 1$:

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

where $\mathsf{dk}_\rho \leftarrow\!\!\$\ \mathsf{RKGen}(\mathsf{msk}, \rho)$, and $\mathsf{dk}_\mathbb{S} \leftarrow\!\!\$\ \mathsf{PolGen}(\mathsf{msk}, \mathbb{S})$.

Definition 15 (CPA-1-sided and CPA-2-sided security of ME)**.** Let $t \in \{1, 2\}$. We say that an ME Π is CPA-t-sided secure if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

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

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

$(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{RKGen}(\mathsf{msk},\cdot),\mathsf{PolGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk})$

$b \leftarrow\!\!\$\ \{0,1\}, c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, \sigma_b, \mathbb{R}_b, m_b)$

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

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

Else **return** 0

Figure 7: Games defining CPA-t-sided security of ME.

where game $\mathbf{G}^{\mathsf{CPA\text{-}t\text{-}ME}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Figure 7. 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):**

$$\begin{aligned}&(\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho) = 0) \vee (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1) = 0)\\ &\quad \vee (\mathbb{R}_0(\rho) = \mathbb{S}(\sigma_1) = 0) \vee (\mathbb{R}_1(\rho) = \mathbb{S}(\sigma_0) = 0);\end{aligned} \tag{7}$$

- **Case $t = 2$ (mismatch and match):** Either

$$\begin{aligned}&(\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho) = 0) \vee (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1) = 0)\\ &\quad \vee (\mathbb{R}_0(\rho) = \mathbb{S}(\sigma_1) = 0) \vee (\mathbb{R}_1(\rho) = \mathbb{S}(\sigma_0) = 0)\end{aligned}$$

$$\text{or } (\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho)) \wedge (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1)) \wedge (m_0 = m_1). \tag{8}$$

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. [AFNV19, Definition 5]. We chose to change their names to avoid confusion and make the notation consistent w.r.t. to the one of PE.

A.2 ME from 2-Key PE

Construction 4. *Let* $\mathsf{keyPE} = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *be a 2-key PE scheme with message space* $\mathcal{M}$*, input space* $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1$*, and predicate space* $\mathcal{P} = \{\mathbb{P}_{\rho,\mathbb{R}}(x_0, x_1)\}_{(\rho,\mathbb{R})\in\mathcal{V}}$ *indexed by* $\mathcal{V} = \mathcal{V}_0 \times \mathcal{V}_1$ *such that*

$$\mathbb{P}_{\rho,\mathbb{R}}(\sigma, \mathbb{S}) = \mathbb{P}_\rho(\mathbb{S}) \wedge \mathbb{P}_\mathbb{R}(\sigma) = \mathbb{S}(\rho) \wedge \mathbb{R}(\sigma),$$

where $\sigma \in \mathcal{X}_0$*,* $\mathbb{S} \in \mathcal{X}_1$*,* $\rho \in \mathcal{V}_0$*, and* $\mathbb{R} \in \mathcal{V}_1$*. We build an ME scheme with message space* $\mathcal{M}$*, sender's policy and attribute spaces* $\mathcal{X}_1$ *and* $\mathcal{X}_0$*, and receiver's policy and attribute spaces* $\mathcal{V}_1$ *and* $\mathcal{V}_0$*, 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* $\mathsf{msk} = (\mathsf{msk}_0, \mathsf{msk}_1)$ *where* $(\mathsf{mpk}, \mathsf{msk}_0, \mathsf{msk}_1) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$*.*

$\mathsf{RKGen}(\mathsf{msk}, \rho)$**:** *Upon input the master secret key* $\mathsf{msk} = (\mathsf{msk}_0, \mathsf{msk}_1)$ *and attributes* $\rho \in \mathcal{V}_0$*, the randomized receiver key generator outputs* $\mathsf{dk}_\rho \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}_0, \rho)$*.*

$\mathsf{PolGen}(\mathsf{msk}, \mathbb{S})$**:** *Upon input the master secret key* $\mathsf{msk} = (\mathsf{msk}_0, \mathsf{msk}_1)$ *and a policy* $\mathbb{S} \in \mathcal{V}_1$*, the randomized receiver policy generator outputs* $\mathsf{dk}_\mathbb{S} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}_1, \mathbb{S})$*.*

$\mathsf{Enc}(\mathsf{mpk}, \sigma, \mathbb{R}, m)$**:** *Upon input the master public key* mpk*, attributes* $\sigma \in \mathcal{X}_0$*, a policy* $\mathbb{R} \in \mathcal{X}_1$*, 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* $\rho \in \mathcal{V}_0$*, a secret decryption key* $\mathsf{dk}_\mathbb{S}$ *for a policy* $\mathbb{S} \in \mathcal{V}_1$*, 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 keyPE. Below, we establish the following result.

Theorem 7. *Let* keyPE *be as above.*

1. *If* keyPE *is CPA-1-sided secure (Definition 11) then the ME scheme* Π *from Construction 4 is CPA-1-sided secure (Definition 15).*
2. *If* keyPE *is CPA-2-sided secure (Definition 11) then the ME scheme* Π *from Construction 4 is CPA-2-sided secure (Definition 15).*

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 keyPE. 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 $\rho \in \mathcal{V}_0$ for RKGen, forward the query ρ to $\mathsf{KGen}(\mathsf{msk}_0, \cdot)$ and return the answer dk_ρ.
   - On input $\mathbb{R} \in \mathcal{V}_1$ from PolGen, forward the query $\mathbb{R}$ to $\mathsf{KGen}(\mathsf{msk}_1, \cdot)$ 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 $i \in \{0, 1\}$. 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 Equation (7). This implies that $\forall \rho \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_0, \cdot)}, \mathbb{R} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1, \cdot)}$, $\mathbb{P}_{\rho,\mathbb{R}}(\sigma_0, \mathbb{S}_0) = \mathbb{S}_0(\rho) \wedge \mathbb{R}(\sigma_0) = 0$ and $\mathbb{P}_{\rho,\mathbb{R}}(\sigma_1, \mathbb{S}_1) = \mathbb{S}_1(\rho) \wedge \mathbb{R}_1(\sigma) = 0$. Hence, A' is a valid adversary for $\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}keyPE}}_{\mathsf{keyPE},\mathsf{A}'}(\lambda)$. 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 keyPE, i.e., it satisfies Equation (8). This implies that $\forall \rho \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_0, \cdot)}, \mathbb{R} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_1, \cdot)}$, either $\mathbb{P}_{\rho,\mathbb{R}}(\sigma_0, \mathbb{S}_0) = \mathbb{P}_{\rho,\mathbb{R}}(\sigma_1, \mathbb{S}_1) = 0$ or $\mathbb{P}_{\rho,\mathbb{R}}(\sigma_0, \mathbb{S}_0) = \mathbb{P}_{\rho,\mathbb{R}}(\sigma_1, \mathbb{S}_1) \wedge m_0 = m_1$. Hence, A' is a valid adversary for $\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}keyPE}}_{\mathsf{keyPE},\mathsf{A}'}(\lambda)$. This concludes the proof. □

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

$(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^\ell)$, $\mathsf{pub} = (\mathsf{ek}_{n-\ell}, \ldots, \mathsf{ek}_{n-1})$

$(\{(m^i_0, \ldots, m^i_{n-1}), (x^i_0, \ldots, x^i_{n-1})\}_{i\in\{0,1\}}, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0^{\mathsf{KGen}(\mathsf{msk},\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[0,n-\ell-1]}}(1^\lambda, \mathsf{pub})$

$b \leftarrow\!\!\$\ \{0,1\}, c_0 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_0, x^b_0, m^b_0), \ldots, c_{n-1} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_{n-1}, x^b_{n-1}, m^b_{n-1})$

$b' \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot),\{\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)\}_{j\in[0,n-\ell-1]}}(1^\lambda, c_0, \ldots, c_{n-1}, \alpha)$

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

Else **return** 0

Figure 8: Game defining CPA-t-sided security of n-input PE in the ℓ-hybrid setting.

B Relating Multi-key PE and Multi-input PE

B.1 Multi-input PE in the Hybrid Setting

A multi-input PE in the hybrid setting allows to generate (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 $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-\ell-1})$ are kept secret whereas the last ℓ keys $(\mathsf{ek}_{n-\ell}, \ldots, \mathsf{ek}_{n-1})$ are published. Observe that, for $\ell = 0$, the hybrid setting corresponds to the secret key setting (see Section 4.2.1).

Definition 16 (ℓ-Hybrid CPA-1-sided and CPA-2-side security of n-input PE)**.** Let $t \in \{1,2\}$. We say that a n-input PE Π is CPA-t-sided secure in the ℓ-hybrid setting if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$:

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

where game $\mathbf{G}^{\ell\text{-hyb-CPA-1-inputPE}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Figure 8. Let $\mathcal{Q}_i = \{x | \exists (x,m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_i,\cdot,\cdot)}\}$ for $i \in [0, n-\ell-1]$. Adversary A is called valid if $\forall \mathbb{P} \in \mathcal{Q}_{\mathsf{KGen}}$, $\forall d \in \{0,1\}$, $\forall (x'_0, \ldots, x'_{n-1}) \in \mathcal{Q}_0 \cup \{x^d_0\} \times \ldots \times \mathcal{Q}_{n-\ell-1} \cup \{x^d_{n-\ell-1}\} \times \mathcal{X}_{n-\ell} \times \ldots \times \mathcal{X}_{n-1}\}$, we have

$$\textbf{Case } t = 1\textbf{:}\ \mathbb{P}(x'_0, \ldots, x'_{j-1}, x^d_j, x'_{j+1}, \ldots, x'_{n-1}) = 0 \tag{9}$$

$$\begin{aligned}\textbf{Case } t = 2\textbf{:}\ &\text{Either } \mathbb{P}(x'_0, \ldots, x'_{j-1}, x^d_j, x'_{j+1}, \ldots, x'_{n-1}) = 0\\ &\text{or } \mathbb{P}(x'_0, \ldots, x'_{j-1}, x^d_j, x'_{j+1}, \ldots, x'_{n-1}) =\\ &\quad \mathbb{P}(x'_0, \ldots, x'_{j-1}, x^{1-d}_j, x'_{j+1}, \ldots, x'_{n-1}) \wedge\\ &\quad m^d_0 = m^{1-d}_0 \wedge \ldots \wedge m^d_{n-1} = m^{1-d}_{n-1}.\end{aligned} \tag{10}$$

B.2 Multi-key PE from Multi-input PE

In the this section, we build a n-key PE from $(n+1)$-input PE that tolerates 1 public encryption key, i.e., 1-hybrid setting (Definition 16).

Construction 5. *Let* $\mathsf{inputPE} = (\mathsf{Setup}_1, \mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *be a* $(n+1)$*-input PE scheme with message space* $\mathcal{M} = \mathcal{M}_0 \times \ldots \times \mathcal{M}_n$*, input space* $\mathcal{X} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_n$*, and predicate space* $\mathcal{P}_1 = \{\mathbb{P}(x_0, \ldots, x_n)\}$ *such that*

$$\mathbb{P}(x_0, \ldots, x_n) = \mathbb{P}_{x_0,\ldots,x_{n-1}}(x_n),$$

where $x_i \in \mathcal{X}_i$ *for* $i \in [0, n]$*. We build a* n*-key PE scheme with message space* $\mathcal{M}_n$*, input space* $\mathcal{X}_n$*, and predicate space* $\mathcal{P} = \{\mathbb{P}_{v_0,\ldots,v_{n-1}}(X)\}_{(v_0,\ldots,v_{n-1})\in\mathcal{V}}$ *indexed by* $\mathcal{V} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_{n-1}$*, 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* $\mathsf{msk}_0 = (\mathsf{ek}_0, \mathsf{dk}_\mathbb{P}), \ldots, \mathsf{msk}_{n-1} = (\mathsf{ek}_{n-1}, \mathsf{dk}_\mathbb{P})$ *where* $(\mathsf{mpk}', \mathsf{ek}_0, \ldots, \mathsf{ek}_n) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$ *and* $\mathsf{dk}_\mathbb{P} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}', \mathbb{P})$ *for* $\mathbb{P} \in \mathcal{P}_1$*.*

$\mathsf{KGen}(\mathsf{msk}_i, v_i)$**:** *Let* $i \in [0, n-1]$*. Upon input the* i*-th master secret key* $\mathsf{msk}_i = (\mathsf{ek}_i, \mathsf{dk}_\mathbb{P})$*, and the* i*-th predicate's index* $v_i \in \mathcal{X}_i$*, the randomized key generator outputs* $\mathsf{dk}_{v_i} = (c_{v_i}, \mathsf{dk}_\mathbb{P})$ *where* $c_{v_i} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{ek}_i, v_i, \bot)$*.*

$\mathsf{Enc}(\mathsf{mpk}, x, m)$**:** *Upon input the master public key* $\mathsf{mpk} = \mathsf{ek}_n$*, attributes* $x \in \mathcal{X}_0$*, a predicate's input* $x \in \mathcal{X}_n$*, and a message* $m \in \mathcal{M}_n$*, the randomized encryption algorithm computes* $c \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{ek}_n, x, m)$*.*

$\mathsf{Dec}(\mathsf{dk}_{v_0}, \ldots, \mathsf{dk}_{v_{n-1}}, c)$**:** *Upon input* n *secret decryption keys* $\mathsf{dk}_{v_0} = (c_{v_0}, \mathsf{dk}_\mathbb{P}), \ldots, \mathsf{dk}_{v_{n-1}} = (c_{v_{n-1}}, \mathsf{dk}_\mathbb{P})$ *and a ciphertext* c*, the deterministic decryption algorithm outputs* m_n *where* $(m_0, \ldots, m_n) = \mathsf{Dec}_1(\mathsf{dk}_\mathbb{P}, c_{v_0}, \ldots, c_{v_{n-1}}, c)$*.*

Correctness follows from the correctness of $\mathsf{inputPE}$. As for security, we establish the following result.

Theorem 8. *Let* $\mathsf{inputPE}$ *be as above. For* $t \in [1, 2]$*, if* $\mathsf{inputPE}$ *is CPA-*t*-sided secure in the* 1*-hybrid model without collusion (Definition 16) then the* n*-key PE scheme* Π *from Construction 5 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 $\mathsf{KGen}(\mathsf{msk}_0, \cdot), \ldots, \mathsf{KGen}(\mathsf{msk}_{n-1}, \cdot)$ (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 $j \in [0, n-1]$). 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 collusion) of $\mathsf{inputPE}$. A' is defined as follows:

1. Receive ek_n from the challenger and send it to A.
2. Send the query $\mathbb{P}$ (i.e., the single predicate supported by $\mathsf{inputPE}$) to the KGen_1 oracle and receive $\mathsf{dk}_\mathbb{P}$.
3. Initialize $\mathcal{L}_i = \{\emptyset\}$ for $i \in [0, n-1]$. 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 $(v_i, \bot)$ 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 $\mathsf{dk}_{v_i} = (c_{v_i}, \mathsf{dk}_\mathbb{P})$.
4. Receive the challenge (m_0, m_1, x_0, x_1) from A. A' sends the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ where $m_0^i = \ldots = m_{n-1}^i = \bot$, $m_n^i = m_i$, $x_j^i = x_j^{1-i} = \bar{x_j} \leftarrow\!\!\$\ \mathcal{L}_j$, and $x_n^i = x_i$, for $j \in \{0, \ldots, n-1\}$ and $i \in \{0, 1\}$.

5. Receive the challenge ciphertexts $c_0, \ldots, c_n$ and forward c_n 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, A is a valid adversary, i.e., $\forall v_0 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_0,\cdot)}, \ldots, v_{n-1} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{n-1},\cdot)}$, we have $\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 0$. In order to be valid, A' needs to satisfy the condition of Equation (9). First, note that $\mathcal{Q}_i \cup \{x_i^d\} = \mathcal{Q}_i \cup \{x_i^{1-d}\} = \mathcal{Q}_i = \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_i,\cdot)} = \mathcal{L}_i$ 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 $\mathbb{P}$ on an input that depends on the challenge bit d (i.e., the cases captured by Equation (9)) is when A' uses the challenge ciphertext c_n. However, when c_n is used, the validity of A implies that $\forall (v_0, \ldots, v_{n-1}) \in \mathcal{Q}_0 \times \ldots \times \mathcal{Q}_{n-1}$,

$$\mathbb{P}(v_0, \ldots, v_n, x_n^0) = \mathbb{P}(v_0, \ldots, v_n, x_n^1) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 0,$$

where $x_n^i = x_0$ for $i \in \{0, 1\}$. Hence, A' submits only a single query to oracle KGen_1 and is also a valid adversary for $\mathbf{G}^{\ell\text{-hyb-CPA-1-inputPE}}_{\mathsf{inputPE},\mathsf{A}'}(\lambda)$. 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'. By definition A is a valid adversary with respect to the CPA-2-sided security of $\mathsf{inputPE}$, i.e., $\forall v_0 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_0,\cdot)}, \ldots, v_{n-1} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{n-1},\cdot)}$, we have

$$\begin{aligned} &\text{Either } \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = 0 \text{ or} \\ &\quad \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) \wedge m_0 = m_1. \end{aligned}$$

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 $\mathsf{A}'s$ follows by using an similar argument to that of CPA-1-sided security and, in addition, observing that

$$\mathbb{P}(v_0, \ldots, v_n, x_n^0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_1) = \mathbb{P}(v_0, \ldots, v_n, x_n^1),$$

and $m_n^0 = m_0 = m_1 = m_n^1$. This concludes the proof. □

C Missing Proofs

C.1 Proof of Theorem 4

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

$$\begin{aligned} \mathcal{P} &= \{\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0, \ldots, x_{n-1})\}_{(v_0,\ldots,v_{n-1})\in\mathcal{V}} \\ &= \{\mathbb{P}_{v_0}(x_0) \wedge \ldots \wedge \mathbb{P}_{v_{n-1}}(x_{n-1})\}_{(\mathbb{P}_{v_0},\ldots,\mathbb{P}_{v_{n-1}})\in\mathcal{P}_0\times\ldots\times\mathcal{P}_{n-1}}. \end{aligned} \tag{11}$$

Also, consider the validity condition of $\mathbf{G}^{\mathsf{CPA-1-keyPE}}_{\Pi,\mathsf{A}}$ (Definition 11). We can write such a validity condition for the predicate space $\mathcal{P}$ as follows: $\forall v_0 \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_0,\cdot)}, \ldots, v_{n-1} \in \mathcal{Q}_{\mathsf{KGen}(\mathsf{msk}_{n-1},\cdot)}$,

$$\begin{aligned} &\mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0^0, \ldots, x_{n-1}^0) = \mathbb{P}_{v_0,\ldots,v_{n-1}}(x_0^1, \ldots, x_{n-1}^1) = 0 \implies \\ &\big(\mathbb{P}_{v_0}(x_0^0) \wedge \ldots \wedge \mathbb{P}_{v_{n-1}}(x_{n-1}^0)\big) = 0 \ \wedge \ \big(\mathbb{P}_{v_0}(x_0^0) \wedge \ldots \wedge \mathbb{P}_{v_{n-1}}(x_{n-1}^0)\big) = 0, \end{aligned}$$

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

$$\mathbb{P}_{v_{j_0}}(x_{j_0}^0) = 0 \wedge \mathbb{P}_{v_{j_1}}(x_{j_1}^1) = 0. \tag{12}$$

Hence, in order to be valid with respect to $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{CPA\text{-}1\text{-}keyPE}}$, A needs to satisfy the above equation. Let $\mathbf{Validity}_{j_0,j_1}$ the validity condition (as defined in Equation (12)) with respect to some $j_0, j_1 \in [0, n-1]$. By taking into account the above point, the CPA-1-sided security of Construction 1 follows by proving the following lemma.

Lemma 1. *Let $j_0, j_1 \in [0, n-1]$. If both PE_{j_0} and PE_{j_1} are CPA secure (Definition 8) and LOBF is secure (Definition 2), then*

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

Proof. Consider the following hybrid experiments:

$\mathbf{H}_0^b(\lambda)$: This is exactly the experiment $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{CPA\text{-}1\text{-}keyPE}}$ conditioned to the event $\mathbf{Validity}_{j_0,j_1}$ where the challenge bit is b.

$\mathbf{H}_1^b(\lambda)$: Same as $\mathbf{H}_0^b$, except that the challenger computes $c_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_{j_b}(\mathsf{mpk}_{j_b}, x_{j_b}^b, w)$ where $w \leftarrow\!\!\$\ \mathcal{M}_{j_b}$ (instead of $c_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_{j_b}(\mathsf{mpk}_{j_b}, x_{j_b}^b, c_{j_b-1})$.

$\mathbf{H}_2^b(\lambda)$: Same as $\mathbf{H}_1^b$, 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. $\mathbf{H}_0^b(\lambda) \approx_c \mathbf{H}_1^b(\lambda)$.

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 of PE_{j_b}. A is defined as follows:

1. Receive mpk_{j_b} from the challenger.
2. Send $\mathsf{mpk} = (\mathsf{mpk}_0, \ldots, \mathsf{mpk}_{n-1})$ to D where $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow\!\!\$\ \mathsf{Setup}_i(1^\lambda)$ for $i \in [0, n-1] \setminus \{j_b\}$.
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 $\mathbb{P}_{v_i} \in \mathcal{P}_{j_b}$ to KGen_{j_b} and returns the answer dk_{v_i}. Otherwise (if $j_b \neq i$), it returns $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_i, \mathbb{P}_{v_i})$ for $\mathbb{P}_{v_i} \in \mathcal{P}_i$.
4. Receive the challenge $(m_0, m_1, (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ from D.
5. Sample $y \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and set $c_{-1} = y$.
6. For $i \in [0, j_b - 1]$, compute $c_i \leftarrow\!\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x_i^b, c_{i-1})$.
7. Send the challenge $(m_0^*, m_1^*, x_{j_b}^b)$ where $m_0^* = c_{j_b-1}$, $m_1^* \leftarrow\!\!\$\ \mathcal{M}_{j_b}$ and receive the challenge ciphertext c^*.
8. For $i \in [j_b + 1, n-1]$, compute $c_i \leftarrow\!\!\$\ \mathsf{Enc}_i(\mathsf{mpk}_i, x_i^b, c_{i-1})$ where $c_{j_b} = c^*$.

9. Finally, send $c = \widetilde{\mathbb{C}} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_{n-1}}, y, m_b)$ 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 conditioned to the event $\mathbf{Validity}_{j_0,j_1}$, we conclude that $\forall v_{j_0} \in \mathcal{V}_{j_0}, \mathbb{P}_{v_{j_0}}(x_{j_b}^0) = 0$. This implies that $\forall \mathbb{P} \in \mathcal{Q}_{\mathsf{KGen}_{j_b}}$, $\mathbb{P}(x_{j_b}^0) = 0$. 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 $\mathsf{mpk} = (\mathsf{mpk}_1, \ldots, \mathsf{mpk}_{n-1})$ to D where $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow\!\!\$\ \mathsf{Setup}_i(1^\lambda)$ for $i \in [0, n-1]$.
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 $\mathsf{dk}_{v_i} \leftarrow\!\!\$\ \mathsf{KGen}_i(\mathsf{msk}_i, \mathbb{P}_{v_i})$ for $\mathbb{P}_{v_i} \in \mathcal{P}_i$.
3. Receive the challenge $(m_0, m_1, (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ from D.
4. For $i \in [j_b + 1, n-1]$, 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 send $(\mathbb{C}_{c_{n-1}}, m_b)$ 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 to 3. □

(CPA-2-sided security of Π). Consider the validity condition of $\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}keyPE}}_{\Pi,\mathsf{A}}$ (Definition 11). This can be rewritten with respect to the definition of $\mathcal{P}$ (Equation (11)) as follows: $\exists j_0, j_1 \in [0, n-1]$, $\forall v_{j_0} \in \mathcal{V}_{j_0}$, $\forall v_{j_1} \in \mathcal{V}_{j_1}$, $\forall (v_0, \ldots, v_{n-1}) \in \mathcal{V}$,

$$\begin{aligned} &\text{Either } \mathbb{P}_{v_{j_0}}(x_{j_0}^0) = 0 \wedge \mathbb{P}_{v_{j_1}}(x_{j_1}^1) = 0 \\ &\quad \text{or } \mathbb{P}_{v_0}(x_0^0) = \mathbb{P}_{v_0}(x_0^1) \wedge \ldots \wedge \mathbb{P}_{v_{n-1}}(x_{n-1}^0) = \mathbb{P}_{v_{n-1}}(x_{n-1}^1) \end{aligned} \tag{13}$$

Consider the following conditions:

$$\begin{aligned}\mathbf{Validity}_{0,j_0,j_1} &: \forall v_{j_0} \in \mathcal{V}_{j_0}, \forall v_{j_1} \in \mathcal{V}_{j_1}, \\ &\quad \mathbb{P}_{v_{j_0}}(x^0_{j_0}) = 0 \wedge \mathbb{P}_{v_{j_1}}(x^1_{j_1}) = 0 \\ \mathbf{Validity}_1 &: \forall (v_0, \ldots, v_{n-1}) \in \mathcal{V}, \\ &\quad \mathbb{P}_{v_0}(x^0_0) = \mathbb{P}_{v_0}(x^1_0) \wedge \ldots \wedge \mathbb{P}_{v_{n-1}}(x^0_{n-1}) = \mathbb{P}_{v_{n-1}}(x^1_{n-1}).\end{aligned}$$

By leveraging the above validity conditions we can rephrase Equation (13) as follows: $\exists j_0, j_1 \in [0, n-1]$ such that

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

Hence, in order to be valid with respect to $\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}keyPE}}_{\Pi,\mathsf{A}}$, A needs to satisfy the above equation. By taking into account the above point, the CPA-2-sided security of Construction 1 follows by proving the following lemmas.

Lemma 2. *Let $j_0, j_1 \in [0, n-1]$. If both PE_{j_0} and PE_{j_1} are CPA secure (Definition 8) and LOBF is secure (Definition 2), then*

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

Proof. Note that by assumption each PE_i is CPA-2-sided secure (Theorem 4). This implies that each PE_i is CPA secure. Hence, the proof of Lemma 2 is identical to that of Lemma 1. □

Lemma 3. *If each $\mathsf{PE}_0, \ldots, \mathsf{PE}_{n-1}$ are CPA-2-sided secure (Definition 9), then*

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

Proof. Consider the following hybrid experiments:

$\mathbf{H}^b_{-1}(\lambda)$: This is exactly the experiment $\mathbf{G}^{\mathsf{CPA\text{-}2\text{-}keyPE}}_{\Pi,\mathsf{A}}$ conditioned to the event $\mathbf{Validity}_1$ where the challenge bit is b.

$\mathbf{H}^b_i(\lambda)$ for $i \in [0, n-1]$: 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 $i \in [0, n-1]$, $\mathbf{H}^b_{i-1}(\lambda) \approx_c \mathbf{H}^b_i(\lambda)$.*

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 $\mathsf{mpk} = (\mathsf{mpk}_0, \ldots, \mathsf{mpk}_{n-1})$ to D where $(\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow\!\!\$\ \mathsf{Setup}_i(1^\lambda)$ for $j \in [0, n-1] \setminus \{i\}$.
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 $\mathbb{P}_{v_j} \in \mathcal{P}_i$ to KGen_i and returns the answer dk_{v_j}. Otherwise (if $j \neq i$), it returns $\mathsf{dk}_{v_j} \leftarrow\!\!\$\ \mathsf{KGen}_j(\mathsf{msk}_j, \mathbb{P}_{v_j})$ for $\mathbb{P}_{v_j} \in \mathcal{P}_j$.

4. Receive the challenge $(m_0, m_1, (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ from D.
5. Sample $y \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and set $c_{-1} = y$.
6. For $j \in [0, i-1]$, compute $c_j \leftarrow\!\!\$\ \mathsf{Enc}_j(\mathsf{mpk}_j, x_j^{1-b}, c_{j-1})$.
7. Send the challenge $(m_0^*, m_1^*, x_i^b, x_i^{1-b})$ where $m_0^* = m_1^* = c_{i-1}$, and receive the challenge ciphertext c^*.
8. For $j \in [i+1, n-1]$, compute $c_j \leftarrow\!\!\$\ \mathsf{Enc}_j(\mathsf{mpk}_j, x_j^b, c_{j-1})$ where $c_i = c^*$.
9. Finally, send $c = \widetilde{\mathbb{C}} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_{n-1}, \mathsf{mpk}_0, \ldots, \mathsf{mpk}_{n-1}}, y, m_b)$ 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 conditioned to the event $\mathbf{Validity}_1$, we conclude that $\forall v \in \mathcal{V}_i, \mathbb{P}_v(x_i^0) = \mathbb{P}_v(x_i^1)$. This implies that $\forall \mathbb{P} \in \mathcal{Q}_{\mathsf{KGen}_i}, \mathbb{P}(x_i^0) = \mathbb{P}(x_i^1)$. Hence, A is a valid adversary with the same advantage of D. This concludes the proof. □

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

Proof. Conditioned to $\mathbf{Validity}_1$, 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 1 is CPA-2-sided secure.

C.2 Proof of Theorem 5

Consider the predicate space $\mathcal{P}_1 = \{\mathbb{P}(x_0, \ldots, x_{n-1})\} = \{\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})\}$ of Construction 2. Let $\mathbb{P}^* \in \mathcal{P}_1$ be the only predicate for which the adversary will ask the decryption key $\mathsf{dk}_{\mathbb{P}^*}$ during the experiment $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{CPA\text{-}1\text{-}inputPE}}$ (recall that we prove the security of Construction 2 in the scenario without collusion, i.e., $|\mathcal{Q}_{\mathsf{KGen}}| = 1$). Also, consider the validity condition of $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{CPA\text{-}1\text{-}inputPE}}$. We can write such a validity condition with respect to $\mathbb{P}^* \in \mathcal{Q}_{\mathsf{KGen}} = \{\mathbb{P}^*\}$ as follows: $\forall j \in [0, n-1]$, $\forall i \in \{0,1\}$, $\forall (x_0', \ldots, x_{n-1}') \in \mathcal{Q}_0 \cup \{x_0^i\} \times \ldots \times \mathcal{Q}_{n-1} \cup \{x_{n-1}^i\}$,

$$\begin{aligned}&\mathbb{P}^*(x_0', \ldots, x_{j-1}', x_j^i, x_{j+1}', \ldots, x_{n-1}')\\ &= \mathbb{P}_0^*(x_0') \wedge \ldots \wedge \mathbb{P}_{j-1}^*(x_{j-1}') \wedge \mathbb{P}_j^*(x_j^i) \wedge \mathbb{P}_{j+1}^*(x_{j+1}') \wedge \ldots \wedge \mathbb{P}_{n-1}^*(x_{n-1}') = 0,\end{aligned}$$

where $\mathcal{Q}_k = \{x | \exists (x, m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_k, \cdot, \cdot)}\}$ for $k \in [0, n-1]$. The above equation can be rewritten as follows: $\exists j_0, j_1 \in [0, n-1]$, $\forall (x_0', \ldots, x_{n-1}') \in \mathcal{Q}_0 \times \ldots \times \mathcal{Q}_{n-1}$,

$$\begin{aligned}&\big(\big(\mathbb{P}_0^*(x_0^0) = 0 \wedge \ldots \wedge \mathbb{P}_{n-1}^*(x_{n-1}^0) = 0\big) \vee \big(\mathbb{P}_{j_0}^*(x_{j_0}^0) = 0 \wedge \mathbb{P}_{j_0}^*(x_{j_0}') = 0\big)\big) \wedge\\ &\quad\big(\big(\mathbb{P}_0^*(x_0^1) = 0 \wedge \ldots \wedge \mathbb{P}_{n-1}^*(x_{n-1}^1) = 0\big) \vee \big(\mathbb{P}_{j_1}^*(x_{j_1}^1) = 0 \wedge \mathbb{P}_{j_1}^*(x_{j_1}') = 0\big)\big) \qquad (14)\end{aligned}$$

Hence, in order to be valid, A needs to satisfy at least one between the conditions defined by Equation (14). These conditions are defined by the events below: for some $j_0, j_1 \in [0, n-1]$,

$$\begin{aligned}
&\mathbf{Validity}_1 : \\
&\quad \mathbb{P}^*_0(x^0_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^0_{n-1}) = 0 \wedge \mathbb{P}^*_0(x^1_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^1_{n-1}) = 0. \\
&\mathbf{Validity}_{2,j_0,j_1} : \forall x'_{j_0} \in \mathcal{Q}_{j_0}, \forall x'_{j_1} \in \mathcal{Q}_{j_1}, \\
&\quad \mathbb{P}^*_{j_0}(x^0_{j_0}) = 0 \wedge \mathbb{P}^*_{j_0}(x'_{j_0}) = 0 \wedge \mathbb{P}^*_{j_1}(x^1_{j_1}) = 0 \wedge \mathbb{P}^*_{j_1}(x'_{j_1}) = 0. \\
&\mathbf{Validity}_{3,j_0} : \forall x'_{j_0} \in \mathcal{Q}_{j_0}, \\
&\quad \mathbb{P}^*_{j_0}(x^0_{j_0}) = 0 \wedge \mathbb{P}^*_{j_0}(x'_{j_0}) = 0 \wedge \mathbb{P}^*_0(x^1_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^1_{n-1}) = 0. \\
&\mathbf{Validity}_{4,j_1} : \forall x'_{j_1} \in \mathcal{Q}_{j_1}, \\
&\quad \mathbb{P}^*_0(x^0_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^0_{n-1}) = 0 \wedge \mathbb{P}^*_{j_1}(x^1_{j_1}) = 0 \wedge \mathbb{P}^*_{j_1}(x'_{j_1}) = 0.
\end{aligned}$$

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

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

Proof. Consider the following hybrid experiments:

$\mathbf{H}^{b,-1}_0(\lambda)$**:** This is exactly the experiment $\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}inputPE}}_{\Pi,\mathsf{A}}(\lambda)$ conditioned to the event $\mathbf{Validity}_1$ where the challenge bit is b.

$\mathbf{H}^{b,i}_0(\lambda)$**:** Same as $\mathbf{H}^{b,i-1}_0$, except that the challenger changes how it computes the challenger ciphertext c_i. Formally, the value $c^{(1)}_i$ challenge ciphertext $c_i = (\widetilde{\mathbb{C}}_i, c^{(2)}_i)$ is computed as $c^{(1)}_i \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ (instead of $c^{(1)}_i \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_i || \mathsf{k}_{i+1})$) where $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$ (for some function k), $x_i = x^0_i$, and $x_j = x^\star_j$ for $j \in [0, n-1] \setminus \{i\}$. Observe that $c^{(1)}_i$ is computed by fixing $x_i = x^0_i$ (instead of $x_i = x^b_i$), i.e., the input $(x_0, \ldots, x_{n-1})$ used to compute the i-th challenge ciphertext is fixed and does not depend on the challenge bit b.

$\mathbf{H}^{b,-1}_1(\lambda)$**:** Identical to $\mathbf{H}^{b,n-1}_0(\lambda)$.

$\mathbf{H}^{b,i}_1(\lambda)$**:** Same as $\mathbf{H}^{b,i-1}_1$, except that the challenger changes how it computes the challenger ciphertext c_i. Formally, the value $\widetilde{\mathbb{C}}_i$ of challenge ciphertext $c_i = (\widetilde{\mathbb{C}}_i, c^{(2)}_i)$ is simulated by the challenger using the simulator of the lockable obfuscation scheme LOBF, i.e., $\widetilde{\mathbb{C}}_i \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c^{(2)}_i, \mathsf{k}_{i+1}}|}, 1^{|m^b_i|})$.

Claim 6. $\mathbf{H}^{b,i-1}_0(\lambda) \approx_c \mathbf{H}^{b,i}_0(\lambda)$ *for* $i \in [0, n-1]$.

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

1. Receive mpk from the challenger.
2. Computes $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.
3. A answers to the incoming oracle queries as follows:

- On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}}$.
- On input $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ for $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ where $j \in [0, n-1]$, A proceeds as follows:
  (a) Sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$.
  (b) Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $x_j = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.
  (c) 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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$.

4. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.

5. For any $j \in [0, n-1] \setminus \{i\}$, A proceeds as follows:

   Case $j < i$: Sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Run $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.

   Case $j = i$: Send the challenge $(m_0^* = y_i || \mathsf{k}_{i+1}, m_1^* = 0^{s(\lambda)+k(\lambda)}, x_0^* = (x^{*0}_0, \ldots, x^{*0}_{n-1}), x_1^* = (x^{*1}_0, \ldots, x^{*1}_{n-1}))$ where $y_i \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$, $x^{*0}_i = x_i^b$, $x^{*1}_i = x_i^0$, and $x^{*0}_j = x^{*1}_j = x^\star_j$ for $j \in [0, n-1] \setminus \{i\}$. Receive the challenge ciphertext c^* from the challenger. Set $c_i^{(1)} = c^*$.

   Case $j > i$: Sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $x_j = x_j^b$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.

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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m_j^b)$ for any $j \in [0, n-1]$.

7. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}_1^{b,i}(\lambda)$. Moreover, conditioned to the event $\mathbf{Validity}_1$, we know that D asks for a single decryption key $\mathsf{dk}_{\mathbb{P}^*}$ for $\mathbb{P}^*$ and $\mathbb{P}_i^*(x_i^0) = 0 \wedge \mathbb{P}_i^*(x_i^1) = 0$. Because of this, A submits a single query $\mathbb{P}^*$ 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* $i \in [0, n-1]$.

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. Computes $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.

2. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P}^*)$.
   - On input $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ for $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ where $j \in [0, n-1]$, A computes $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j\}$. 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 $\widetilde{\mathbb{C}}_1 \leftarrow\!\!\$\ \mathsf{Obf}(\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$.
3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ from D.
4. For any $j \in [0, n-1]$, compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ and $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$ where $x_j = x_j^0$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.
5. For any $j \in [0, n-1] \setminus \{i\}$, A proceeds as follows:

   Case $j < i$: Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^b|})$.

   Case $j = i$: Send the challenge $(\mathbb{C}_{c_i^{(2)}, \mathsf{k}_{i+1}}, m_i^b)$ to the challenger and receive $\widetilde{\mathbb{C}}$. Set $\widetilde{\mathbb{C}}_i = \widetilde{\mathbb{C}}$.

   Case $j > i$: Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m_j^b)$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$.
6. Set $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ for $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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. $\mathbf{H}_1^{b,n-1}(\lambda) \equiv \mathbf{H}_1^{1-b,n-1}(\lambda)$.

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

By combining Claims 6 to 8 and conditioned to the event $\mathbf{Validity}_1$, we conclude that

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

This concludes the proof. □

Lemma 5. *Let $j_0, j_1 \in [0, n-1]$. If PE is CPA-1-sided secure without collusion (Definition 8), SKE is CPA secure (Definition 4), and LOBF is secure (Definition 2), then*

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

Proof. Without loss of generality, let $q = |\mathcal{Q}_0| = \ldots = |\mathcal{Q}_{n-1}| \in \mathsf{poly}(\lambda)$. Consider the following hybrid experiments:

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

$\mathbf{H}_1^b(\lambda)$**:** 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 $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ (instead of $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_{j_b}||\mathsf{k}_{j_b+1})$) 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 $j \in [0, n-1] \setminus \{j_b\}$.

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

$\mathbf{H}_2^{b,i}(\lambda)$**:** 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 $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_{j_b} = x$, and $x_j = x_j^\star$ for $j \in [0, n-1] \setminus \{j_b\}$. Finally, the challenger returns $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}, c_{j_b}^{(2)})$ where $c_{j_b}^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_{j_b}, c_{j_b}^{(1)})$, $y_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, and $\widetilde{\mathbb{C}}_{j_b} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}, y_{j_b}, m)$. 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., $\widetilde{\mathbb{C}}_{j_b} \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}|}, 1^{|m_{j_b}^b|})$.

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

$\mathbf{H}_4^{b,i}(\lambda)$**:** 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., $\widetilde{\mathbb{C}}_{j_b} \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}|}, 1^{|m|})$. Otherwise, on input the i'-th query (x, m) such that $i' > i$, the challenger answers as usual, i.e., as defined in $\mathbf{H}_4^{b,0}$.

$\mathbf{H}_5^{b,q,q,1}$**:** Identical to $\mathbf{H}_4^{b,q}(\lambda)$.

$\mathbf{H}_{5+i}^{b,0,0,0}$**:** 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 $v \equiv j_b + i \mod n$. Formally, the value $c_v^{(1)}$ is computed as $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ 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 $j \in [0, n-1] \setminus \{v\}$. 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's input $(x_0, \ldots, x_{n-1})$ used to compute the v-th challenge ciphertext is fixed and does not depend on the challenge bit b.

$\mathbf{H}_{5+i}^{b,t_1,0,0}$**:** 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 $v \equiv j_b + i \mod n$. Formally, on input the t_1'-th query (x, m) such that $t_1' \leq t_1$, the challenger computes $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, and $x_j = x_j^\star$ for $j \in [0, n-1] \setminus \{v\}$. Finally, the challenger returns $c_v = (\widetilde{\mathbb{C}}_v, c_v^{(2)})$ where $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, y_v, m)$, and $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. 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 $\mathbf{H}_{5+i}^{b,0,0,0}$.

$\mathbf{H}^{b,q,t_2,0}_{5+i}$**:** Same as $\mathbf{H}^{b,q,t_2-1,0}_{5+i}(\lambda)$ except that the challenger changes how it answers to the first t_2 queries for oracle $\mathsf{Enc}(\mathsf{ek}_v,\cdot,\cdot)$ where $v \equiv j_b + i \mod n$. 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., $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_v^{(2)},\mathsf{k}_{v+1}}|}, 1^{|m|})$. 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 $\mathbf{H}^{b,q,0,0}_{5+i}$.

$\mathbf{H}^{b,q,q,1}_{5+i}$**:** Same as $\mathbf{H}^{b,q,q,0}_{5+i}(\lambda)$ except that the challenger changes how it computes the challenger ciphertext c_v where $v \equiv j_b + i \mod n$. 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., $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_v^{(2)},\mathsf{k}_{v+1}}|}, 1^{|m_v^b|})$.

Claim 9. $\mathbf{H}^b_0(\lambda) \approx_c \mathbf{H}^b_1(\lambda)$.

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-1-sided security without collusion of PE. A is defined as follows:

1. Receive mpk from the challenger.
2. Computes $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.
3. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}^*}$.
   - On input $(x,m) \in \mathcal{X}_1 \times \mathcal{M}_3$ for $\mathsf{Enc}(\mathsf{ek}_j,\cdot,\cdot)$ where $j \in [0, n-1]$, A proceeds as follows:
     (a) Sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$.
     (b) Compute $c_j^{(j)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0,\ldots,x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $x_j = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.
     (c) 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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)},\mathsf{k}_{j+1}}, y_j, m)$.
4. Receive the challenge $((m_0^0,\ldots,m_{n-1}^0),(m_0^1,\ldots,m_{n-1}^1),(x_0^0,\ldots,x_{n-1}^0),(x_0^1,\ldots,x_{n-1}^1)$ from D. Send the challenge $(m_0^* = y_{j_b}||\mathsf{k}_{j_b+1}, m_1^* = 0^{s(\lambda)+k(\lambda)}, x_0^* = (x^{*0}_0,\ldots,x^{*0}_{n-1}), x_1^* = (x^{*1}_0,\ldots,x^{*1}_{n-1}))$ where $y_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $0^{s(\lambda)+k(\lambda)} \in \mathcal{M}_1$, $x^{*0}_{j_b} = x^{*1}_{j_b} = x^b_{j_b}$ and $x^{*0}_j = x^{*1}_j = x^\star_j$ for $j \in [0, n-1] \setminus \{j_b\}$.
5. Receive the challenge ciphertext c^* from the challenger. Set $c_{j_b}^{(1)} = c^*$.
6. For any $j \in [0, n-1] \setminus \{j_b\}$, compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0,\ldots,x_{n-1}), y_j||\mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x_j^b$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.
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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)},\mathsf{k}_{j+1}}, y_j, m_j^b)$ for any $j \in [0, n-1]$.
8. Send the challenge ciphertexts $(c_0,\ldots,c_{n-1})$ 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, D submits a single query $\mathbb{P}^*$ to oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$ and, conditioned to the event $\mathbf{Validity}_{2,j_0,j_1}$, we know that $\mathbb{P}^*_{j_b}(x^b_{j_b}) = 0$. Because of this, A submits only a query to oracle $\mathsf{KGen}_1(\mathsf{msk}, \cdot)$ (i.e., security without collusion) 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 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 collusion of PE. A is defined as follows:

1. Receive mpk from the challenger.

2. Computes $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.

3. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}^*}$.
   - On input i'-th query $(x, m) \in \mathcal{X}_1 \times \mathcal{M}_3$ for $\mathsf{Enc}(\mathsf{ek}_j, \cdot, \cdot)$ where $j \in [0, n-1]$, A proceeds as follows:

     Case $j \neq j_b$: Sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $x_j = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.

     Case $j = j_b$ and $i' < i$: Sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_{j_b} = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$.

     Case $j = j_b$ and $i' = i$: Sample $y_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Send the challenge $(m_0^* = y_{j_b} || \mathsf{k}_{j_b+1}, m_1^* = 0^{s(\lambda)+k(\lambda)}, x_0^* = ({x^*}^0_0, \ldots, {x^*}^0_{n-1}), x_1^* = ({x^*}^1_0, \ldots, {x^*}^1_{n-1}))$ to the challenger where ${x^*}^0_{j_b} = {x^*}^1_{j_b} = x$ and ${x^*}^0_{j'} = {x^*}^1_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$. Receive the challenge ciphertext c^* and $c_{j_b}^{(1)} = c^*$.

     Case $j = j_b$ and $i' > i$: Sample $y_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Compute $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_{j_b} || \mathsf{k}_{j_b+1})$ where $x_{j_b} = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$.

     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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$.

4. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.

5. For every $j \in [0, n-1] \setminus \{j_b\}$, sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $x_j = x_j^b$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.

6. Sample $y_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and compute the ciphertext $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_{j_b} = x_{j_b}^b$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$.

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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)},\mathsf{k}_{j+1}}, y_j, m_j^b)$ for any $j \in [0, n-1]$.
8. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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, we know that D submits a single query $\mathbb{P}^*$ to oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$ and, conditioned to the event $\mathbf{Validity}_{2,j_0,j_1}$, we know that $\forall x'_{j_b} \in \mathcal{Q}_{j_b}, \mathbb{P}^*_{j_b}(x'_{j_b}) = 0$. 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. $\mathbf{H}_2^{b,q}(\lambda) \approx_c \mathbf{H}_3^b(\lambda)$.

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. A is defined as follows:

1. Computes $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.
2. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P}^*)$.
   - 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 $y_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Compute $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_{j_b} = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j_b\}$.

     Case $j \neq j_b$: Run $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j\}$.

     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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)},\mathsf{k}_{j+1}}, y_j, m)$.
3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ from D.
4. Compute $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ 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 $j \in [0, n-1] \setminus \{j_b\}$.
5. For any $j \in [0, n-1] \setminus \{j_b\}$, sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)},\mathsf{k}_{j+1}}, y_j, m_j^b)$ where $x_j = x_j^b$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.
6. Send the challenge $(\mathbb{C}_{c_{j_b}^{(2)},\mathsf{k}_{j_b+1}}, m_{j_b}^b)$ to the challenger and receive $\widetilde{\mathbb{C}}$. Set $\widetilde{\mathbb{C}}_{j_b} = \widetilde{\mathbb{C}}$.
7. Set $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ for $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}_2^{b,q}(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}_3^{b}(\lambda)$. Hence, A has the same advantage of D. This concludes the proof. □

Claim 12. $\mathbf{H}_4^{b,i-1}(\lambda) \approx_c \mathbf{H}_4^{b,i}(\lambda)$.

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. Computes $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.

2. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P}^*)$.
   - 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 $i' < i$: Compute $\widetilde{\mathbb{C}}_{j_b} \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}|}, 1^{|m|})$, $c_{j_b}^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_{j_b}, c_{j_b}^{(1)})$, and $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_{j_b} = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j_b\}$.

     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 $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_{j_b} = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j_b\}$. Send the challenge $(\mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}, m)$ 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 $\widetilde{\mathbb{C}}_{j_b} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}, y_{j_b}, m)$, $c_{j_b}^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_{j_b}, c_{j_b}^{(1)})$, $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $y_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_{j_b} = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j_b\}$.

     Case $j \neq j_b$: Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j\}$.

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

3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ from D.

4. Compute $\widetilde{\mathbb{C}}_{j_b} \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}|}, 1^{|m_{j_b}^b|})$, $c_{j_b}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ 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 $j \in [0, n-1] \setminus \{j_b\}$.

5. For any $j \in [0, n-1] \setminus \{j_b\}$, sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m_j^b)$ where $x_j = x_j^b$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.

6. Set $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ for $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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. $\mathbf{H}_{5+i-1}^{b,q,q,1}(\lambda) \approx_c \mathbf{H}_{5+i}^{b,0,0,0}(\lambda)$ *for* $i \in [n]$.

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

1. Computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$ and $\mathsf{ek}_j = (\mathsf{ek}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ for $j \in [0, n-1] \setminus \{v\}$. Let $\mathsf{k}_n = \mathsf{k}_0$.
2. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P}^*)$.
   - 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 $j \in \{j_b, j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{j\}$.

     Case $j = v$**:** Run $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_v || \mathsf{k}_{v+1})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{v\}$. 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 $i < n-1$**,** $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$, the ciphertext $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{j\}$.

     Finally, return $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$.
3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.
4. **Case** $i < n$ **(hence,** $v \neq j_b$**):** For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j \in \{j_b, j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$**:** Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^b$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^b|})$ and $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

   Case $j = v$**:** Run $c_v^{(1,0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x^{*0}_0, \ldots, x^{*0}_{n-1}), y_v || \mathsf{k}_{v+1})$ and $c_v^{(1,1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x^{*1}_0, \ldots, x^{*1}_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x^{*0}_v = x_v^b$, $x^{*1}_v = x_v^0$, and $x^{*0}_{j'} = x^{*1}_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{v\}$. 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 $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_v}, y_v, m_v^b)$.

Case $i < n-1$, $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$**:** Run $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x_j^b$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_j}, y_j, m_j^b)$ and $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

5. **Otherwise, case** $i = n$ **(hence,** $v = j_b$**):** For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j \in \{j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$**:** Execute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^b$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^b|})$ and $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

   Case $j = v = j_b$**:** Compute $c_{j_b}^{(1,0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x^{*0}_0, \ldots, x^{*0}_{n-1}), 0^{s(\lambda)+k(\lambda)})$ and $c_{j_b}^{(1,1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x^{*1}_0, \ldots, x^{*1}_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x^{*0}_{j_b} = x^b_{j_b}$, $x^{*1}_{j_b} = x^0_{j_b}$, and $x^{*0}_{j'} = x^{*1}_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$. 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 $\widetilde{\mathbb{C}}_{j_b} \leftarrow\!\!\$\ \widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_{j_b}^{(2)}, \mathsf{k}_{j_b+1}}|}, 1^{|m_{j_b}^b|})$.

6. Set $c_j = (\widetilde{\mathbb{C}}_j, c_j^{(2)})$ for $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}_{5+i-1}^{b,q,q,1}(\lambda)$. 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* $i \in [n-1]$.

Proof. Let $v \equiv j_b + i \mod n$ 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. Computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$ and $\mathsf{ek}_j = (\mathsf{ek}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ for $j \in \{0, \ldots, n-1\} \setminus \{v\}$. Let $\mathsf{k}_n = \mathsf{k}_0$.

2. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P}^*)$.
   - 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 $j \in \{j_b, j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{j\}$.

     Case $j = v$ **and** $t_1' < t_1$**:** Sample $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Run $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{v\}$. 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 $j = v$ and $t'_1 = t_1$: Run $c_v^{(1,0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_v || \mathsf{k}_{v+1})$ and $c_v^{(1,1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{v\}$. 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 $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_v}, y_v, m)$.

Case $j = v$ and $t'_1 > t_1$: Sample $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Run $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), k_v || \mathsf{k}_{v+1})$ where $x_v = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{v\}$. 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 $i < n-1$, $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$: Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j\}$.

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

3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.

4. For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j \in \{j_b, j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$: Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^b|})$ and $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

   Case $j = v$: Sample $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)+k(\lambda)}$ and compute $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x_v^0$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{v\}$. 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 $i < n-1$, $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$: Run $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x_j^b$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_j}, y_j, m_j^b)$ 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 $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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* $i \in [n-1]$.

Proof. Let $v \equiv j_b + i \mod n$. 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. Computes $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, k_{j+1})$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.

2. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P}^*)$.
   - 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 $j \in \{j_b, j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j\}$.

     Case $j = v$ **and** $t_2' < t_2$**:** Compute $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}|}, 1^{|m|})$, $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, and $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{v\}$.

     Case $j = v$ **and** $t_2' = t_2$**:** Compute $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$ and $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{v\}$. Send the challenge $(\mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, m)$ 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 $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. Compute $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{v\}$. 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 $i < n-1$**,** $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j\}$.

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

3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.

4. For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j \in \{j_b, j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$**:** Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^b|})$ and $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$.

   Case $j = v$**:** Sample $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)+k(\lambda)}$ and compute $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x_v^0$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{v\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, y_v, m_v^b)$ and $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$.

   Case $i < n-1$**,** $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$**:** Run $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x_j^b$, and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_j}, y_j, m_j^b)$ 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 $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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. $\mathbf{H}_{5+i}^{b,q,t_2-1,0}(\lambda) \approx_c \mathbf{H}_{5+i}^{b,q,t_2,0}(\lambda)$ *for* $t_2 \in [q]$ *and* $i \in [n-1]$.

Proof. Let $v \equiv j_b + i \mod n$. 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. Computes $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{k}_j, k_{j+1})$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.

2. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P}^*)$.
   - 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 $j \in \{j_b, j_b + 1 \mod n, \ldots, v \equiv j_b + i \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{j\}$.

     Case $i < n-1$**,** $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{j\}$.

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

3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.

4. For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j \in \{j_b, j_b + 1 \mod n, \ldots, j_b + i - 1 \mod n\}$**:** Compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^b|})$ 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 $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x_v^0$ and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{v\}$. Send the challenge $(\mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, m_v^b)$ to the challenger and receive the answer $\widetilde{\mathbb{C}}^*$. Set $\widetilde{\mathbb{C}}_v = \widetilde{\mathbb{C}}^*$.

   Case $i < n-1$**,** $j \in \{j_b + i + 1 \mod n, \ldots, j_b + n - 1 \mod n\}$**:** Run $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x_j^b$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_j}, y_j, m_j^b)$ 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 $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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. $\mathbf{H}_{5+n}^{1-b,q,q,q}(\lambda) \approx_c \mathbf{H}_{5+n}^{b,q,q,1}(\lambda)$.

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

By combining Claims 9 to 17 and conditioned to the event $\mathbf{Validity}_{2,j_0,j_1}$, we conclude that

$$\begin{aligned}\mathbf{H}_0^b \approx_c \mathbf{H}_1^b \equiv \mathbf{H}_2^{b,0} \approx_c \ldots \approx_c \mathbf{H}_2^{b,q} \approx_c \mathbf{H}_3^b \equiv \mathbf{H}_4^{b,0} \approx_c \ldots \approx_c \mathbf{H}_4^{b,q} \equiv \\ \mathbf{H}_5^{b,q,q,1} \approx_c \mathbf{H}_6^{b,0,0,0} \approx_c \ldots \approx_c \mathbf{H}_6^{b,q,0,0} \approx_c \ldots \approx_c \mathbf{H}_6^{b,q,q,0} \approx_c \\ \mathbf{H}_6^{b,q,q,1} \approx_c \ldots \approx_c \mathbf{H}_{5+n}^{b,0,0,0} \approx_c \mathbf{H}_{5+n}^{1-b,0,0,0}.\end{aligned}$$

This concludes the proof. □

Lemma 6. *Let $j_0 \in [0, n-1]$. If PE is CPA-1-sided secure without collusion (Definition 8), SKE is CPA secure (Definition 4), and LOBF is secure (Definition 2), then*

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

Proof. Without loss of generality, let $q = |\mathcal{Q}_0| = \ldots = |\mathcal{Q}_{n-1}| \in \mathsf{poly}(\lambda)$. Consider the hybrid experiments of Lemma 4 and Lemma 5. Formally,

- Let $\mathbf{H}_0^{1,-1}(\lambda), \mathbf{H}_0^{1,i}(\lambda), \mathbf{H}_1^{1,-1}(\lambda), \mathbf{H}_1^{1,i}(\lambda)$, for $i \in [0, n-1]$, be the hybrid 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 $\mathbf{H}_0^0(\lambda), \mathbf{H}_1^0(\lambda), \mathbf{H}_2^{0,i}(\lambda), \mathbf{H}_3^0(\lambda), \mathbf{H}_4^{0,i}(\lambda), \mathbf{H}_5^{0,q,q,1}(\lambda), \mathbf{H}_{5+j}^{0,i,0,0}(\lambda), \mathbf{H}_{5+j}^{0,q,i,0}(\lambda), \mathbf{H}_{5+j}^{0,q,q,k}(\lambda)$, and $\mathbf{H}_{5+n}^{0,0,0,0}(\lambda)$, for $(i,j,k) \in [0,q] \times [1,n-1] \times \{0,1\}$, be the hybrid 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:

$\mathbf{H}_{6+n}^{0,q,q}$**:** Identical to $\mathbf{H}_{5+n}^{0,0,0,0}$.

$\mathbf{H}_{6+n+i}^{0,0,0}$**:** Identical to $\mathbf{H}_{6+n+i-1}^{0,q,q}$.

$\mathbf{H}_{6+n+i}^{0,0,t_2}$**:** Same as $\mathbf{H}_{6+n+i}^{0,0,t_2-1}$ except that the challenger changes how it answers to the first t_2 queries for oracle $\mathsf{Enc}(\mathsf{ek}_v, \cdot, \cdot)$ where $v \equiv j_0 - i \mod n$. 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 $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)},\mathsf{k}_{v+1}}, y_v, m)$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$. 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 $\mathbf{H}_{6+n+i}^{0,0,0}$.

$\mathbf{H}^{0,t_1,q}_{6+n+i}$**:** Same as $\mathbf{H}^{0,t_1-1,q}_{6+n+i}$ except that the challenger changes how it answers to the first t_1 queries for oracle $\mathsf{Enc}(\mathsf{ek}_v,\cdot,\cdot)$ where $v \equiv j_0 - i \mod n$. Formally, on input the t'_1-th query (x,m) such that $t'_1 \leq t_1$, the challenger computes $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_0,\ldots,x_{n-1}),y_v||\mathsf{k}_{v+1})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x$, and $x_j = x_j^\star$ for $j \in [0,n-1]\setminus\{v\}$. Finally, the challenger returns $c_v = (\widetilde{\mathbb{C}}_v, c_v^{(2)})$ where $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)},\mathsf{k}_{v+1}}, y_v, m)$. 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 $\mathbf{H}^{0,0,q}_{6+n+i}$.

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

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

Claim 19. $\mathbf{H}^{0,0,t_2-1}_{6+n+i}(\lambda) \approx_c \mathbf{H}^{0,0,t_2}_{6+n+i}(\lambda)$ *for* $t_2 \in [q]$ *and* $i \in [n]$.

Proof. Let $v \equiv j_0 - i \mod n$. Suppose there exists a PPT distinguisher D that distinguishes between $\mathbf{H}^{0,0,t_2-1}_{6+n+i}(\lambda)$ and $\mathbf{H}^{0,0,t_2}_{6+n+i}(\lambda)$ 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. Computes $(\mathsf{ek}_0,\ldots,\mathsf{ek}_{n-1},\mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk},\mathsf{k}_j,k_{j+1})$ for $j \in [0,n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.

2. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk},\mathbb{P}^*)$.
   - 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** $j \in \{j_0 - i + 1 \mod n,\ldots,j_0 - 1 \mod n\}$**:** Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)},\mathsf{k}_{j+1}}, y_j, m)$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_0,\ldots,x_{n-1}),y_j||\mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0,n-1]\setminus\{j\}$.

     Case $j = v$ **and** $t'_2 < t_2$**:** Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)},\mathsf{k}_{v+1}}, y_v, m)$, $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, and $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_0,\ldots,x_{n-1}),0^{s(\lambda)+k(\lambda)})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0,n-1]\setminus\{v\}$.

     Case $j = v$ **and** $t'_2 = t_2$**:** Compute $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$ and $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_0,\ldots,x_{n-1}),0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0,n-1]\setminus\{v\}$. 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** $t'_2 > t_2$**:** Compute $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_v^{(2)},\mathsf{k}_{v+1}}|}, 1^{|m|})$, $c_v^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_v, c_v^{(1)})$, and $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_0,\ldots,x_{n-1}),0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0,n-1]\setminus\{v\}$.

     Case $i \neq n$ **and** $j \in \{j_0 - i - 1 \mod n,\ldots,j_0\}$**:** Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)},\mathsf{k}_{j+1}}|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk},(x_0,\ldots,x_{n-1}),0^{s(\lambda)+k(\lambda)})$ where $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0,n-1]\setminus\{j\}$.

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

3. Receive the challenge $((m_0^0,\ldots,m_{n-1}^0),(m_0^1,\ldots,m_{n-1}^1),(x_0^0,\ldots,x_{n-1}^0),(x_0^1,\ldots,x_{n-1}^1))$ from D.

4. For every $j \in [0, n-1]$, the adversary A computes $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^0|})$ 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 $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}_{6+n+i}^{0,0,t_2}(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}_{6+n+i}^{0,0,t_2-1}(\lambda)$. Hence, A has the same advantage of D. This concludes the proof. □

Claim 20. $\mathbf{H}_{6+n+i}^{0,t_1-1,q}(\lambda) \approx_c \mathbf{H}_{6+n+i}^{0,t_1,q}(\lambda)$ *for* $t_1 \in [q]$ *and* $i \in [n-1]$.

Proof. Let $v \equiv j_0 - i \mod n$. Suppose there exists a PPT distinguisher D that distinguishes between $\mathbf{H}_{6+n+i}^{0,t_1,q}(\lambda)$ and $\mathbf{H}_{6+n+i}^{0,t_1-1,q}(\lambda)$ with non-negligible probability. We build an adversary A that breaks the CPA security of SKE. A is defined as follows:

1. Computes $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}_1(1^\lambda)$ and $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{ek}_j, \mathsf{ek}_{j-1})$ for $j \in [0, n-1] \setminus \{v\}$. Let $\mathsf{k}_n = \mathsf{k}_0$.

2. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P}^*)$.
   - 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 $i > 1$, $j \in \{j_0 - i + 1 \mod n, \ldots, j_0 - 1 \mod n\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}, y_j, m)$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{j\}$.

     Case $j = v$, $t_1' < t_1$**:** Compute $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_v || \mathsf{k}_{v+1})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{v\}$. Send the query $c_v^{(1)}$ to the oracle Enc_2 and receive the answer $c_v^{(2)}$. Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, y_v, m)$.

     Case $j = v$, $t_1' = t_1$**:** Compute $c_v^{(1,0)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ and $c_v^{(1,1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_v || \mathsf{k}_{v+1})$ where $y_v \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$, $x_v = x$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{v\}$. 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 $\widetilde{\mathbb{C}}_v \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_v}, y_v, m)$.

     Case $j = v$, $t_1' > t_1$**:** Run $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{v\}$. Send the query $c_v^{(1)}$ to the oracle Enc_2 and receive the answer $c_v^{(2)}$. Compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c_v^{(2)}, \mathsf{k}_{v+1}}, y_v, m)$ where $y_v go \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$.

     Case $j \in \{j_0 - i - 1 \mod n, \ldots, j_0\}$**:** Run $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m|})$, $c_j^{(2)} \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{k}_j, c_j^{(1)})$, and $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{j\}$.

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

3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.

4. For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j = v$: Computes $c_v^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_v = x_v^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{v\}$. Send the query $c_v^{(1)}$ to the oracle Enc_2 and receive the answer $c_v^{(2)}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^0|})$.

   Case $j \neq v$: Computes $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$. Finally, compute $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c_j^{(2)}, \mathsf{k}_{j+1}}|}, 1^{|m_j^0|})$ 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 $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}_{6+n+i}^{0,t_1-1,q}(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}_{6+n+i}^{0,t_1,q}(\lambda)$. Hence, A has the same advantage of D. This concludes the proof. □

Claim 21. $\mathbf{H}_{6+2n}^{0,t_1-1,q}(\lambda) \approx_c \mathbf{H}_{6+2n}^{0,t_1,q}(\lambda)$ *for* $t_1 \in [q]$.

Proof. Suppose there exists a PPT distinguisher D that distinguishes between $\mathbf{H}_{6+2n}^{0,t_1,q}(\lambda)$ and $\mathbf{H}_{6+2n}^{0,t_1-1,q}(\lambda)$ with non-negligible probability. We build an adversary A that breaks the CPA-1-sided security without collusion of PE. A is defined as follows:

1. Receive mpk from the challenger.

2. Computes $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_2(1^\lambda)$ for $j \in [0, n-1]$. Let $\mathsf{k}_n = \mathsf{k}_0$.

3. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}^*}$.
   - 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 $j \in [0, n-1]$, A proceeds as follows:

     Case $j \neq j_0$: Sample $y_j \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and compute $c_j^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j || \mathsf{k}_{j+1})$ where $x_j = x$ and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$.

     Case $j = j_0$ and $t_1' < t_1$: Compute $c_{j_0}^{(1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_{j_0} || \mathsf{k}_{j_0+1})$ where $x_{j_0} = x$ and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j_0\}$.

     Case $j = j_0$ and $t_1' = t_1$: Sample $y_{j_0} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and send the challenge $(m_0^* = 0^{s(\lambda)+k(\lambda)}, m_1^* = y_{j_0} || \mathsf{k}_{j_0+1}, x_0^* = (x^{*0}{}_0, \ldots, x^{*0}{}_{n-1}), x_1^* = (x^{*1}{}_0, \ldots, x^{*1}{}_{n-1}))$ to the challenger where $x^{*0}{}_{j_0} = x^{*1}{}_{j_0} = x$ and $x^{*0}{}_{j'} = x^{*1}{}_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j_0\}$. Receive the challenge ciphertext c^* and $c_{j_0}^{(1)} = c^*$.

Case $j = j_0$ and $t'_1 > t_1$: Sample $y_{j_0} \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$ and compute $c^{(1)}_{j_0} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_{j_0} = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_0\}$.

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 $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, \mathbb{C}_{c^{(2)}_j, \mathsf{k}_{j+1}}, y_j, m)$.

4. Receive the challenge $((m^0_0, \ldots, m^0_{n-1}), (m^1_0, \ldots, m^1_{n-1}), (x^0_0, \ldots, x^0_{n-1}), (x^1_0, \ldots, x^1_{n-1})$ from D.

5. For every $j \in [0, n-1]$, the adversary A computes $c^{(1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s(\lambda)+k(\lambda)})$ where $x_j = x^0_j$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.

6. For every $j \in [0, n-1]$, the adversary A computes $c^{(2)}_j \leftarrow\!\!\$\ \mathsf{Enc}_2(\mathsf{ek}_j, c^{(1)}_j)$ and $\widetilde{\mathbb{C}}_j \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}_{c^{(2)}_j, \mathsf{k}_{j+1}}|}, 1^{|m^0_j|})$.

7. Set $c_j = (\widetilde{\mathbb{C}}_j, c^{(2)}_j)$ for $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}^{0,t_1-1,q}_{6+2n}(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}^{0,t_1,q}_{6+2n}(\lambda)$. Moreover, D submits a single query $\mathcal{P}^*$ to oracle $\mathsf{KGen}(\mathsf{msk}, \cdot)$ and, conditioned to the event $\mathbf{Validity}_{3,j_0}$, we know that $\forall x'_{j_0} \in \mathcal{Q}_{j_0}, \mathbb{P}^*_{j_0}(x'_{j_0}) = 0$. 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 $\mathbf{G}^{\mathsf{CPA\text{-}1\text{-}PE}}_{\mathsf{PE},\mathsf{A}}(\lambda)$ with the same advantage of D. This concludes the proof. □

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

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

Claim 23. $\mathbf{H}^{0,q,q}_{6+2n}(\lambda) \equiv \mathbf{H}^{1,q}_1(\lambda)$.

Proof. Claim 23 follows by observing that experiments $\mathbf{H}^{0,q,q}_{6+2n}(\lambda)$ and $\mathbf{H}^{1,q}_1(\lambda)$ are identical (and does not depend on the bit b). □

By combining Claims 18 to 23 and conditioned to the event $\mathbf{Validity}_{3,j_0,}$, we conclude that

$$\mathbf{H}^0_0 \approx_c \mathbf{H}^{0,0,0,0}_{5+n} \equiv \mathbf{H}^{b,q,q}_{6+n} \equiv \mathbf{H}^{b,0,0}_{6+n+1} \approx_c \ldots \approx_c \mathbf{H}^{b,0,q}_{6+n+1} \approx_c \ldots \approx_c \mathbf{H}^{b,q,q}_{6+n+1} \equiv \mathbf{H}^{b,0,0}_{6+n+2} \approx_c \ldots \approx_c \mathbf{H}^{b,0,q}_{6+2n} \approx_c \ldots \approx_c \mathbf{H}^{b,q,q}_{6+2n} \equiv \mathbf{H}^{1,q}_1 \approx_c \mathbf{H}^{1,-1}_0$$

This concludes the proof. □

Lemma 7. *Let $j_1 \in \{0, \ldots, n-1\}$. If PE is CPA-1-sided-secure without collusions (Definition 8), SKE is CPA-secure (Definition 4), and LOBF is secure (Definition 2), then*

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

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

By combining Lemmas 4 to 7 we conclude that Π is CPA-1-sided secure without collusion.

C.3 Proof of Theorem 6

Consider the predicate space $\mathcal{P}_1 = \{\mathbb{P}(x_0, \ldots, x_{n-1})\} = \{\mathbb{P}_0(x_0) \wedge \ldots \wedge \mathbb{P}_{n-1}(x_{n-1})\}$ of Construction 3. Let $\mathbb{P}^* \in \mathcal{P}_1$ be the only predicate for which the adversary will ask the decryption key $\mathsf{dk}_{\mathbb{P}^*}$ during the experiment $\mathbf{G}^{(n-1)\text{-CPA-1-inputPE}}_{\Pi,\mathsf{A}}$ (recall that we prove the security of Construction 3 in the ℓ-corruption setting without collusion (i.e., $|\mathcal{Q}_{\mathsf{KGen}}| = 1$). Also, consider the validity condition of $\mathbf{G}^{(n-1)\text{-CPA-1-inputPE}}_{\Pi,\mathsf{A}}$. We can write such a validity condition with respect to $\mathbb{P}^* \in \mathcal{Q}_{\mathsf{KGen}} = \{\mathbb{P}^*\}$ as follows: $\forall j \in [0, n-1]$, $\forall i \in \{0,1\}$, $\forall (x'_0, \ldots, x'_{n-1}) \in \mathcal{Q}_0 \cup \{x^i_0\} \times \ldots \times \mathcal{Q}_{n-1} \cup \{x^i_{n-1}\}$,

$$\begin{aligned}&\mathbb{P}^*(x'_0, \ldots, x'_{j-1}, x^i_j, x'_{j+1}, \ldots, x'_{n-1})\\ &= \mathbb{P}^*_0(x'_0) \wedge \ldots \wedge \mathbb{P}^*_{j-1}(x'_{j-1}) \wedge \mathbb{P}^*_j(x^i_j) \wedge \mathbb{P}^*_{j+1}(x'_{j+1}) \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x'_{n-1}) = 0,\end{aligned}$$

where $\mathcal{Q}_k = \{x | \exists (x, m) \in \mathcal{Q}_{\mathsf{Enc}(\mathsf{ek}_k, \cdot, \cdot)}\}$ for $k \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$, and $\mathcal{Q}_k = \mathcal{X}_{1,k}$ for $k \in \mathcal{Q}_{\mathsf{Corr}}$ (recall that $|\mathcal{Q}_{\mathsf{Corr}}| \leq n-1$). Note that Construction 3 has input space $\mathcal{X}_1 = \mathcal{X}_{1,0} \times \ldots \times \mathcal{X}_{1,n-1}$ (that is identical to the one of the underlying PE). Hence, we can conclude that for each $\mathcal{X}_{1,i}$ for $i \in [0, n-1]$ there exists $x^\star_i \in \mathcal{X}_{1,i}$ such that $\mathbb{P}^*_i(x^\star_i) = 1$. As a consequence, an adversary is valid only if there exists $j_0, j_1 \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$ such that $\mathbb{P}^*_{j_0}(x^0_{j_0}) = \mathbb{P}^*_{j_1}(x^1_{j_1}) = 0$. 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 i-th predicate's wildcard $x^\star_i \in \mathcal{X}_{1,i}$ for $i \in \mathcal{Q}_{\mathsf{Corr}}$.

According to the above observation, the A's validity can be rewritten as follows: $\exists j_0, j_1 \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$, $\forall (x'_0, \ldots, x'_{n-1}) \in \mathcal{Q}_0 \times \ldots \times \mathcal{Q}_{n-1}$,

$$\begin{aligned}&\Big(\big(\mathbb{P}^*_0(x^0_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^0_{n-1}) = 0\big) \vee \big(\mathbb{P}^*_{j_0}(x^0_{j_0}) = 0 \wedge \mathbb{P}^*_{j_0}(x'_{j_0}) = 0\big)\Big) \wedge\\ &\quad\Big(\big(\mathbb{P}^*_0(x^1_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^1_{n-1}) = 0\big) \vee \big(\mathbb{P}^*_{j_1}(x^1_{j_1}) = 0 \wedge \mathbb{P}^*_{j_1}(x'_{j_1}) = 0\big)\Big)\end{aligned} \tag{15}$$

Hence, in order to be valid, A needs to satisfy at least one between the conditions defined by Equation (15). These conditions are defined by the events below: For some $j_0, j_1 \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$,[10]

$\mathbf{Validity}_1$:
$\mathbb{P}^*_0(x^0_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^0_{n-1}) = 0 \wedge \mathbb{P}^*_0(x^1_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^1_{n-1}) = 0.$

$\mathbf{Validity}_{2,j_0,j_1}$: $\forall x'_{j_0} \in \mathcal{Q}_{j_0}, \forall x'_{j_1} \in \mathcal{Q}_{j_1}$,
$\mathbb{P}^*_{j_0}(x^0_{j_0}) = 0 \wedge \mathbb{P}^*_{j_0}(x'_{j_0}) = 0 \wedge \mathbb{P}^*_{j_1}(x^1_{j_1}) = 0 \wedge \mathbb{P}^*_{j_1}(x'_{j_1}) = 0.$

$\mathbf{Validity}_{3,j_0}$: $\forall x'_{j_0} \in \mathcal{Q}_{j_0}$,
$\mathbb{P}^*_{j_0}(x^0_{j_0}) = 0 \wedge \mathbb{P}^*_{j_0}(x'_{j_0}) = 0 \wedge \mathbb{P}^*_0(x^1_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^1_{n-1}) = 0.$

$\mathbf{Validity}_{4,j_1}$: $\forall x'_{j_1} \in \mathcal{Q}_{j_1}$,
$\mathbb{P}^*_0(x^0_0) = 0 \wedge \ldots \wedge \mathbb{P}^*_{n-1}(x^0_{n-1}) = 0 \wedge \mathbb{P}^*_{j_1}(x^1_{j_1}) = 0 \wedge \mathbb{P}^*_{j_1}(x'_{j_1}) = 0.$

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

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

Proof. Consider the following hybrid experiments:

[10] Since we are in the $(n-1)$-corruptions setting (i.e, $|\mathcal{Q}_{\mathsf{Corr}}| \leq n-1$) such as $j_0, j_1 \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$ always exist.

$\mathbf{H}_0^{b,-1}(\lambda)$**:** This is exactly the experiment $\mathbf{G}_{\Pi,\mathsf{A}}^{(n-1)\text{-}\mathsf{CPA}\text{-}1\text{-}\mathsf{inputPE}}(\lambda)$ conditioned to the validity event $\mathbf{Validity}_1$ where the challenge bit is b.

$\mathbf{H}_0^{b,i}(\lambda)$**:** Same as $\mathbf{H}_0^{b,i-1}$, except that the challenger changes how it computes the challenger ciphertext c_i. Formally, it computes value $c_i^{(-1)} \leftarrow\!\!\$\ c_i^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1})$ (instead of $c_i^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_i^{\mathsf{in}} || y_i^{\mathsf{out}})$) where $c_i^{(-1)}$ is the value used to compute the challenge ciphertext $x_i = x_i^0$, and $x_j = x_j^\star$ for $j \in [0, n-1] \setminus \{i\}$. Observe that c_i is computed by fixing $x_i = x_i^0$ (instead of $x_i = x_i^b$), i.e., the predicate's input $(x_0, \ldots, x_{n-1})$ used to compute the i-th challenge ciphertext is fixed and does not depend on the challenge bit b.

$\mathbf{H}_1^{b,-1}(\lambda)$**:** Identical to $\mathbf{H}_0^{b,n-1}(\lambda)$.

$\mathbf{H}_1^{b,i}(\lambda)$**:** Same as $\mathbf{H}_1^{b,i-1}$, except that the challenger changes how it computes the challenger ciphertext c_i. Formally, the value $\widetilde{\mathbb{C}}_i^{\mathsf{in}}$ 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 Sim_3 of the lockable obfuscation scheme LOBF_3, i.e., $\widetilde{\mathbb{C}}_i^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_i^{(n-1)}, \mathsf{sk}_i, i}|}, 1^{|(\mathsf{sk}_i, i)|})$.

$\mathbf{H}_2^{b,-1}(\lambda)$**:** Identical to $\mathbf{H}_1^{b,n-1}(\lambda)$.

$\mathbf{H}_2^{b,i}(\lambda)$**:** Same as $\mathbf{H}_2^{b,i-1}$, except that the challenger changes how it computes the challenger ciphertext c_i. Formally, 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 Sim_4 of the lockable obfuscation scheme LOBF_4, i.e., $\widetilde{\mathbb{C}}_i^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_i^{(n-1)}, \mathsf{sk}_i, i}|}, 1^{|m_i^b|})$.

Claim 24. $\mathbf{H}_0^{b,i-1}(\lambda) \approx_c \mathbf{H}_0^{b,i}(\lambda)$ *for* $i \in [0, n-1]$.

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 collusion 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 $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$ for $j \in [0, n-1]$.
3. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}^*}$.
   - On input $j \in [0, n-1]$ 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 $j \in [0, n-1]$, return $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
4. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.
5. For any $j \in [0, n-1]$, A proceeds as follows:

   Case $j < i$**:** Sample $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Compute $c_j^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$.

Case $j = i$: Send the challenge $(m_0^* = y_i^{\mathsf{in}} || y_i^{\mathsf{out}}, m_1^* = 0^{s_3(\lambda)+s_4(\lambda)}, x^* = (x^*{}_0, \ldots, x^*{}_{n-1}))$ where $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$, $x^*{}_i = x_i^b$, and $x^*{}_j = x_j^\star$ for $j \in [0, n-1] \setminus \{i\}$. Receive the challenge ciphertext c^* from the challenger. Set $c_i^{(-1)} = c^*$.

Case $j > i$: Sample $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Compute $c_j^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j^{\mathsf{in}} || y_j^{\mathsf{out}})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$.

6. For every $j \in [0, n-1]$, compute $c_j^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ for $v \in [0, n-1]$.
7. Compute $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}, y_j^{\mathsf{in}}, (\mathsf{sk}_j, j))$ and $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}, y_j^{\mathsf{out}}, m_j^b)$ for any $j \in [0, n-1]$.
8. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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, conditioned to the event $\mathbf{Validity}_1$, we know that D asks for a single decryption key $\mathsf{dk}_{\mathbb{P}^*}$ for $\mathbb{P}^*$ and $\mathbb{P}_i^*(x_i^0) = 0 \wedge \mathbb{P}_i^*(x_i^1) = 0$. Because of this, A submits a single query $\mathbb{P}^*$ 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 25. $\mathbf{H}_1^{b,i-1}(\lambda) \approx_c \mathbf{H}_1^{b,i}(\lambda)$ *for* $i \in [0, n-1]$.

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. Computes $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$ for $j \in [0, n-1]$.
2. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P}^*)$.
   - On input $j \in [0, n-1]$ 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 $j \in [0, n-1]$, return $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.
4. For every $j \in [0, n-1]$, run $c_j^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$.
5. For every $j \in [0, n-1]$, compute $c_j^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ for $v \in [0, n-1]$.
6. For any $j \in [0, n-1]$, A proceeds as follows:

   Case $j < i$: Compute $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|(\mathsf{sk}_j, j)|})$.

Case $j = i$: Send the challenge $(\mathbb{C}^{\mathsf{in}}_{c_i^{(n-1)},\mathsf{sk}_i,i}, (\mathsf{sk}_i, i))$ to the challenger and receive $\widetilde{\mathbb{C}}^{\mathsf{in}}_i$.

Case $j > i$: Compute $\widetilde{\mathbb{C}}^{\mathsf{in}}_j \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)},\mathsf{sk}_j,j}, y^{\mathsf{in}}_j, (\mathsf{sk}_j, j))$ where $y^{\mathsf{in}}_j \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)}$.

7. For every $j \in [0, n-1]$, compute $\widetilde{\mathbb{C}}^{\mathsf{out}}_j \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_j^{(n-1)},\mathsf{sk}_j,j}, y^{\mathsf{out}}_j, m^b_j)$ where $y^{\mathsf{out}}_j \leftarrow\!\!\$\ \{0,1\}^{s_4(\lambda)}$.
8. Set $c_j = (\widetilde{\mathbb{C}}^{\mathsf{in}}_j, \widetilde{\mathbb{C}}^{\mathsf{out}}_j)$ for $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 26. $\mathbf{H}_2^{b,i-1}(\lambda) \approx_c \mathbf{H}_2^{b,i}(\lambda)$ *for* $i \in [0, n-1]$.

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. Computes $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$ for $j \in [0, n-1]$.
2. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P}^*)$.
   - On input $j \in [0, n-1]$ 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 $j \in [0, n-1]$, return $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.
4. For every $j \in \{i, \ldots, n-1\}$, compute $c_j^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_j = x_j^0$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$.
5. For every $j \in \{i, \ldots, n-1\}$, run $c_j^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ for $v \in [0, n-1]$.
6. For every $j \in [0, n-1]$, compute $\widetilde{\mathbb{C}}^{\mathsf{in}}_j \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)},\mathsf{sk}_j,j}|}, 1^{|(\mathsf{sk}_j,j)|})$.
7. For every $j \in [0, n-1]$, A proceeds as follows:

   Case $j < i$: Compute $\widetilde{\mathbb{C}}^{\mathsf{out}}_j \leftarrow\!\!\$\ \mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)},\mathsf{sk}_j,j}|}, 1^{|m_j^b|})$.

   Case $j = i$: Send the challenge $(\mathbb{C}^{\mathsf{out}}_{c_i^{(n-1)},\mathsf{sk}_i,i}, m_i^b)$ to the challenger and receive $\widetilde{\mathbb{C}}^{\mathsf{out}}_i$.

   Case $j > i$: Compute $\widetilde{\mathbb{C}}^{\mathsf{out}}_j \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)},\mathsf{sk}_j,j}, y^{\mathsf{out}}_j, m^b_j)$ where $y^{\mathsf{out}}_j \leftarrow\!\!\$\ \{0,1\}^{s_4(\lambda)}$.

8. Set $c_j = (\widetilde{\mathbb{C}}^{\mathsf{in}}_j, \widetilde{\mathbb{C}}^{\mathsf{out}}_j)$ for $j \in [0, n-1]$ and send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 27. $\mathbf{H}_2^{b,n-1}(\lambda) \equiv \mathbf{H}_2^{1-b,n-1}(\lambda)$.

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

By combining Claims 24 to 27 and conditioned to the event $\mathbf{Validity}_1$, we conclude that

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

This concludes the proof. □

Lemma 9. *Let $j_0, j_1 \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$. If PE is CPA secure without collusion (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*

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

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

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

$\mathbf{H}_1^b(\lambda)$: Same as $\mathbf{H}_0^b$, except that the challenger changes how it computes the challenge j_b-th ciphertext c_{j_b}. Formally, it computes $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ (instead of $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_i^{\mathsf{in}} || y_i^{\mathsf{out}})$) where the value $c_{j_b}^{(-1)}$ is used to compute the challenge ciphertext, $x_i = x_i^b$, and $x_j = x_j^\star$ for $j \in [0, n-1] \setminus \{j_b\}$.

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

$\mathbf{H}_2^{b,i}(\lambda)$: 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 $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_{j_b} = x$, and $x_j = x_j^\star$ for $j \in [0, n-1] \setminus \{j_b\}$. Finally, the challenger returns $c_{j_b} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b})$ where $c_{j_b}^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_b}^{(b-1)})$ for $v \in [0, n-1]$, $(y^{\mathsf{in}}_{j_b}, y^{\mathsf{out}}_{j_b}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$, $\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y^{\mathsf{in}}_{j_b}, (\mathsf{sk}_{j_b}, j_b))$, and $\widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y^{\mathsf{out}}_{j_b}, m^b_{j_b})$. Otherwise, on input the i'-th query (x, m) such that $i' > i$, the challenger answers as usual, i.e., as defined in $\mathbf{H}_2^{b,0}$.

$\mathbf{H}_3^b(\lambda)$**:** 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 Sim_3 of the lockable obfuscation scheme LOBF_3, i.e., $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}$ is computes by executing $\mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_{j_b}^{(n-1)},\mathsf{sk}_{j_b},j_b}|}, 1^{|(\mathsf{sk}_{j_b},j_b)|})$.

$\mathbf{H}_4^b(\lambda)$**:** 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 Sim_4 of the lockable obfuscation scheme LOBF_4, i.e., $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}}$ is computed by executing $\mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_{j_b}^{(n-1)},\mathsf{sk}_{j_b},j_b}|}, 1^{|m^b_{j_b}|})$.

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

$\mathbf{H}_5^{b,i}(\lambda)$**:** 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 Sim_3 of the lockable obfuscator scheme LOBF_3, i.e., $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_{j_b}^{(n-1)},\mathsf{sk}_{j_b},j_b}|}, 1^{|(\mathsf{sk}_{j_b},j_b)|})$. 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 $\mathbf{H}_5^{b,q}(\lambda)$.

$\mathbf{H}_6^{b,i}(\lambda)$**:** 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 Sim_4 of the lockable obfuscator scheme LOBF_4, i.e., $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_{j_b}^{(n-1)},\mathsf{sk}_{j_b},j_b}|}, 1^{|m|})$. Otherwise, on input the i'-th query (x, m) such that $i' > i$, the challenger answers as usual, i.e., as defined in $\mathbf{H}_6^{b,0}$.

$\mathbf{H}_7^{b,1,1}$**:** Identical to $\mathbf{H}_6^{b,q}(\lambda)$.

$\mathbf{H}_{7+i}^{b,0,0}$**:** Same as $\mathbf{H}_{7+i-1}^{b,1,1}$ except that the challenger changes how it computes the challenge ciphertext c_r where $r \equiv j_b + i \mod n$. 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 $w \leftarrow\!\!\$\ \mathcal{M}_{2,j_b}$.

$\mathbf{H}_{7+i}^{b,1,0}$**:** Same as $\mathbf{H}_{7+i}^{b,1,0}(\lambda)$ except that the challenger changes how it computes the challenge ciphertext c_r where $r \equiv j_b + i \mod n$. 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 $\mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_r^{(n-1)},\mathsf{sk}_r,r}|}, 1^{|(\mathsf{sk}_r,r)|})$.

$\mathbf{H}_{7+i}^{b,1,1}$**:** Same as $\mathbf{H}_{7+i}^{b,1,0}(\lambda)$ except that the challenger changes how it computes the challenge ciphertext c_r where $r \equiv j_b + i \mod n$. 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 $\mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_r^{(n-1)},\mathsf{sk}_r,r}|}, 1^{|m^b_r|})$.

Claim 28. $\mathbf{H}_0^b(\lambda) \approx_c \mathbf{H}_1^b(\lambda)$.

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 collusion 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 $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$ for $j \in [0, n-1]$.
3. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}^*}$.
   - On input $j \in [0, n-1]$ 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 $j \in [0, n-1]$, return $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x, m)$.
4. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D. Send the challenge $(m_0^* = y_{j_b}^{\mathsf{in}} || y_{j_b}^{\mathsf{out}}, m_1^* = 0^{s_3(\lambda)+s_4(\lambda)}, x^* = ({x^*}_0, \ldots, {x^*}_{n-1}))$ where $(y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$, ${x^*}_{j_b} = x_{j_b}^b$ and ${x^*}_j = x_j^\star$ for $j \in [0, n-1] \setminus \{j_b\}$.
5. Receive the challenge ciphertext c^* from the challenger. Set $c_{j_b}^{(-1)} = c^*$.
6. For every $j \in [0, n-1] \setminus \{j_b\}$, compute $c_j^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j^{\mathsf{in}} || y_j^{\mathsf{out}})$ where $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$, $x_j = x_j^b$, and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j\}$.
7. For every $j \in [0, n-1]$, compute $c_j^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_j^{(v-1)})$ for $v \in [0, n-1]$.
8. Compute $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}_{c_j^{(n-1)}, \mathsf{sk}_j, j}, y_j^{\mathsf{in}}, (\mathsf{sk}_j, j))$ and $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}_{c_j^{(n-1)}, \mathsf{sk}_j, j}, y_j^{\mathsf{out}}, m_j^b)$ for any $j \in [0, n-1]$.
9. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbb{P}^*$ to oracle KGen and, conditioned to the event $\mathbf{Validity}_{2,j_0,j_1}$, we know that $\mathbb{P}^*_{j_b}(x_{j_b}^b) = 0$. 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 29. $\mathbf{H}_2^{b,i-1}(\lambda) \approx_c \mathbf{H}_2^{b,i}(\lambda)$ *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 collusion 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 $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$ for $j \in [0, n-1]$.

3. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}^*}$.
   - On input $j \in [0, n-1]$ 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 $j \in [0, n-1]$, A proceeds as follows:

     Case $j \neq j_b$: Sample $(y_j^{\mathsf{in}} y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Compute $c_j^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j^{\mathsf{in}} || y_j^{\mathsf{out}})$ where $x_j = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j\}$.

     Case $j = j_b$ and $i' < i$: Sample $(y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Compute $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_{j_b} = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$.

     Case $j = j_b$ and $i' = i$: Sample $(y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Send the challenge $(m_0^* = y_{j_b}^{\mathsf{in}} || y_{j_b}^{\mathsf{out}}, m_1^* = 0^{s_3(\lambda)+s_4(\lambda)}, x^* = (x^*{}_0, \ldots, x^*{}_{n-1}))$ to the challenger where $x^*{}_{j_b} = x$ and $x^*{}_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$. Receive the challenge ciphertext c^* and set $c_{j_b}^{(-1)} = c^*$.

     Case $j = j_b$ and $i' > i$: Sample $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Compute $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_j^{\mathsf{in}} || y_j^{\mathsf{out}})$ where $x_{j_b} = x$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$.

     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 $v \in [0, n-1]$, $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}, y_j^{\mathsf{in}}, (\mathsf{sk}_j, j))$ and $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}, y_j^{\mathsf{out}}, m)$.

4. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.

5. Run $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $(y_{j_b}^{\mathsf{in}}, y_{j_b}^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$, $x_{j_b} = x_{j_b}^b$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$.

6. Compute $c_{j_b}^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_b}^{(v-1)})$ for $v \in [0, n-1]$.

7. Compute $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y_{j_b}^{\mathsf{in}}, (\mathsf{sk}_{j_b}, j_b))$ and $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y_{j_b}^{\mathsf{out}}, m_{j_b}^b)$.

8. For every $j \in [0, n-1] \setminus \{j_b\}$, compute $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x_j^b, m_j^b)$.

9. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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, we know that D submits a single query $\mathbb{P}^*$ to oracle KGen and, conditioned

to the event $\mathbf{Validity}_{2,j_0,j_1}$, we know that $j_b \notin \mathcal{Q}_{\mathsf{Corr}}$ and $\forall x'_{j_b} \in \mathcal{Q}_{j_b} \subset \mathcal{X}_{1,j_b}, \mathbb{P}^*_{j_b}(x'_{j_b}) = 0$. Because of this, A submits a single query to oracle KGen_1 and it is also a valid adversary for the experiment $\mathbf{G}^{\mathsf{CPA\text{-}PE}}_{\mathsf{PE},\mathsf{A}}(\lambda)$ with the same advantage of D. This concludes the proof. □

Claim 30. $\mathbf{H}^{b,q}_2(\lambda) \approx_c \mathbf{H}^b_3(\lambda)$.

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_3. A is defined as follows:

1. Compute $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ where $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$ for $j \in [0, n-1]$.

2. A answers to the incoming oracle queries as follows:

   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \mathbb{P}^*)$.
   - On input $j \in [0, n-1]$ 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 $(y^{\mathsf{in}}_{j_b}, y^{\mathsf{out}}_{j_b}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Compute $c^{(-1)}_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_{j_b} = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j_b\}$.

     Case $j \neq j_b$: Compute $c^{(-1)}_j \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y^{\mathsf{in}}_j || y^{\mathsf{out}}_j)$ where $(y^{\mathsf{in}}_j, y^{\mathsf{out}}_j) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$, $x_j = x$, $x_{j'} = x^\star_{j'}$ for any $j' \in [0, n-1] \setminus \{j\}$.

     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 $v \in [0, n-1]$, $\widetilde{\mathbb{C}}^{\mathsf{in}}_j \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c^{(n-1)}_j, \mathsf{sk}_j, j}, y^{\mathsf{in}}_j, (\mathsf{sk}_j, j))$ and $\widetilde{\mathbb{C}}^{\mathsf{out}}_j \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c^{(n-1)}_j, \mathsf{sk}_j, j}, y^{\mathsf{out}}_j, m)$.

3. Receive the challenge $((m^0_0, \ldots, m^0_{n-1}), (m^1_0, \ldots, m^1_{n-1}), (x^0_0, \ldots, x^0_{n-1}), (x^1_0, \ldots, x^1_{n-1})$ from D.

4. Compute $c^{(-1)}_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_{j_b} = x^b_{j_b}$ and $x_{j'} = x^\star_{j'}$ for $j' \in [0, n-1] \setminus \{j_b\}$.

5. Compute $c^{(v)}_{j_b} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c^{(v-1)}_{j_b})$ for $v \in [0, n-1]$.

6. Send the challenge $(\mathbb{C}^{\mathsf{in}}_{c^{(n-1)}_{j_b}, \mathsf{sk}_{j_b}, j_b}, (\mathsf{sk}_{j_b}, j_b))$ to the challenger and receive $\widetilde{\mathbb{C}}$. Set $c_{j_b} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b})$ where $\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_b} = \widetilde{\mathbb{C}}$, $\widetilde{\mathbb{C}}^{\mathsf{out}}_{j_b} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c^{(n-1)}_{j_b}, \mathsf{sk}_{j_b}, j_b}, y^{\mathsf{out}}_{j_b}, m^b_{j_b})$ and $y^{\mathsf{out}}_{j_b} \leftarrow\!\!\$\ \{0,1\}^{s_4(\lambda)}$.

7. For every $j \in [0, n-1] \setminus \{j_b\}$, compute $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x^b_j, m^b_j)$.

8. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 31. $\mathbf{H}_4^{b,i-1}(\lambda) \approx_c \mathbf{H}_4^{b,i}(\lambda)$.

Proof. Claim 31 follows by leveraging a similar argument to that of Claim 30. □

Claim 32. $\mathbf{H}_5^{b,i-1}(\lambda) \approx_c \mathbf{H}_5^{b,i}(\lambda)$ *for* $i \in [q]$.

Proof. Suppose there exists a PPT distinguisher D that distinguishes between $\mathbf{H}_5^{b,i-1}(\lambda)$ and $\mathbf{H}_5^{b,i}(\lambda)$ 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 $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$.
2. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, forward the query $\mathbb{P}^*$ to KGen_1 and return the answer $\mathsf{dk}_{\mathbb{P}^*}$.
   - On input $j \in [0, n-1]$ 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 $j \in [0, n-1]$, A proceeds as follows:

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

     Case $j = j_b$ and $i' < i$: Sample $y_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \{0,1\}^{s_4(\lambda)}$. Compute $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_{j_b} = x$ and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j_b\}$. 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 $v \in [0, n-1]$, $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}|}, 1^{|(\mathsf{sk}_{j_b}, j_b)|})$, and $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y_{j_b}^{\mathsf{out}}, m)$.

     Case $j = j_b$ and $i' = i$: Sample $y_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \{0,1\}^{s_4(\lambda)}$. Compute $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_{j_b} = x$ and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j_b\}$. Send the challenge $(\mathbb{C}^{\mathsf{in}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, (\mathsf{sk}_{j_b}, j_b))$ to the challenger where $c_{j_b}^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_{j_b}^{(v-1)})$ for $v \in [0, n-1]$. Receive the challenge ciphertext $\widetilde{\mathbb{C}}$ and set $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} = \widetilde{\mathbb{C}}$. Return $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y_{j_b}^{\mathsf{out}}, m)$.

     Case $j = j_b$ and $i' > i$: Sample $(y_j^{\mathsf{in}}, y_j^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$. Compute $c_{j_b}^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), 0^{s_3(\lambda)+s_4(\lambda)})$ where $x_{j_b} = x$ and $x_{j'} = x_{j'}^\star$ for $j' \in [0, n-1] \setminus \{j_b\}$. 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 $v \in [0, n-1]$, $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y_{j_b}^{\mathsf{in}}, (\mathsf{sk}_{j_b}, j_b))$, and $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}, y_{j_b}^{\mathsf{out}}, m)$.
3. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1))$ from D.
4. Compute $c_{j_b} = (\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}}, \widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}|}, 1^{|(\mathsf{sk}_{j_b}, j_b)|})$ and $\widetilde{\mathbb{C}}_{j_b}^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_{j_b}^{(n-1)}, \mathsf{sk}_{j_b}, j_b}|}, 1^{|m_{j_b}^b|})$.
5. For every $j \in [0, n-1] \setminus \{j_b\}$, compute $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x_j^b, m_j^b)$.

6. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 33. $\mathbf{H}_6^{b,i-1}(\lambda) \approx_c \mathbf{H}_6^{b,i}(\lambda)$ *for* $i \in [q]$.

Proof. Claim 33 follows by leveraging a similar argument to that of Claim 32. □

Claim 34. $\mathbf{H}_{7+i-1}^{b,1,1}(\lambda) \approx_c \mathbf{H}_{7+i}^{b,0,0}(\lambda)$ *for* $i \in [n-1]$.

Proof. Let $r \equiv j_b + i \mod n$. 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 $j \in [0, n-1] \setminus \{j_b\}$.
2. Receive pk_{j_b} from the challenger.
3. Set $\mathsf{ek}_j = (\mathsf{mpk}, \mathsf{sk}_j, \mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$ for $j \in [0, n-1] \setminus \{j_b\}$.
4. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P}^*)$.
   - On input $j \in [0, n-1]$ 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 $j = j_b$**:** Compute $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|(\mathsf{sk}_j, j)|})$ and $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|m|})$.

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

     Finally, return c_j.
5. Receive the challenge $((m_0^0, \ldots, m_{n-1}^0), (m_0^1, \ldots, m_{n-1}^1), (x_0^0, \ldots, x_{n-1}^0), (x_0^1, \ldots, x_{n-1}^1)$ from D.
6. For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j \in \{j_b, j_b + 1 \mod n, \ldots, r-1 \mod n\}$**:** Run $c_j = (\widetilde{\mathbb{C}}_j^{\mathsf{in}}, \widetilde{\mathbb{C}}_j^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_j^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|(\mathsf{sk}_j, j)|})$ and $\widetilde{\mathbb{C}}_j^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|m_j^b|})$.

   Case $j = r$**:** Sample $(y_r^{\mathsf{in}}, y_r^{\mathsf{out}}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$ and compute $c_r^{(-1)} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0, \ldots, x_{n-1}), y_r^{\mathsf{in}} || y_r^{\mathsf{out}})$ where $x_r = x_r^b$, $x_{j'} = x_{j'}^\star$ for any $j' \in [0, n-1] \setminus \{r\}$. Compute $c_r^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_r^{(v-1)})$ for $v \in \{0, \ldots, j_b - 1\}$. 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 $v \in \{j_b + 1, \ldots, n-1\}$. Set $c_r = (\widetilde{\mathbb{C}}_r^{\mathsf{in}}, \widetilde{\mathbb{C}}_r^{\mathsf{out}})$ where $\widetilde{\mathbb{C}}_r^{\mathsf{in}} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c_r^{(n-1)}, \mathsf{sk}_r, r}, y_r^{\mathsf{in}}, (\mathsf{sk}_r, r))$ and $\widetilde{\mathbb{C}}_r^{\mathsf{out}} \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_r^{(n-1)}, \mathsf{sk}_r, r}, y_r^{\mathsf{out}}, m_r^b)$.

Case $j \in \{r+1 \mod n, \ldots, j_b - 1 \mod n\}$**:** Compute $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x_j^b, m_j^b)$.

7. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}^{b,1,1}_{7+i-1}(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}^{b,0,0}_{7+i}(\lambda)$. Hence, A has the same advantage of D. This concludes the proof. □

Claim 35. $\mathbf{H}^{b,0,0}_{7+i}(\lambda) \approx_c \mathbf{H}^{b,1,0}_{7+i}(\lambda)$ *for* $i \in [n-1]$.

Proof. Let $r \equiv j_b + i \mod n$. Suppose there exists a PPT distinguisher D that distinguishes between $\mathbf{H}^{b,0,0}_{7+i}(\lambda)$ and $\mathbf{H}^{b,1,0}_{7+i}(\lambda)$ 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 $(\mathsf{ek}_0, \ldots, \mathsf{ek}_{n-1}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$.
2. A answers to the incoming oracle queries as follows:
   - On input $\mathbb{P}^* \in \mathcal{P}_1$ for KGen, return $\mathsf{dk}_{\mathbb{P}^*} \leftarrow\!\!\$\ \mathsf{KGen}_1(\mathsf{msk}, \mathbb{P}^*)$.
   - On input $j \in [0, n-1]$ 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 $c_j = (\widetilde{\mathbb{C}}^{\mathsf{in}}_j, \widetilde{\mathbb{C}}^{\mathsf{out}}_j)$ where $\widetilde{\mathbb{C}}^{\mathsf{in}}_j \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|(\mathsf{sk}_j, j)|})$ and $\widetilde{\mathbb{C}}^{\mathsf{out}}_j \leftarrow\!\!\$\ \mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|m|})$.

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

     Finally, return c_j.
3. Receive the challenge $((m^0_0, \ldots, m^0_{n-1}), (m^1_0, \ldots, m^1_{n-1}), (x^0_0, \ldots, x^0_{n-1}), (x^1_0, \ldots, x^1_{n-1})$ from D.
4. For every $j \in [0, n-1]$, the adversary A proceeds as follows:

   Case $j \in \{j_b, j_b + 1 \mod n, \ldots, r-1 \mod n\}$**:** Run $c_j = (\widetilde{\mathbb{C}}^{\mathsf{in}}_j, \widetilde{\mathbb{C}}^{\mathsf{out}}_j)$ where $\widetilde{\mathbb{C}}^{\mathsf{in}}_j \leftarrow\!\!\$\ \mathsf{Sim}_3(1^\lambda, 1^{|\mathbb{C}^{\mathsf{in}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|(\mathsf{sk}_j, j)|})$ and $\widetilde{\mathbb{C}}^{\mathsf{out}}_j \leftarrow\!\!\$\ \mathsf{Sim}_4(1^\lambda, 1^{|\mathbb{C}^{\mathsf{out}}_{c_j^{(n-1)}, \mathsf{sk}_j, j}|}, 1^{|m^b_j|})$.

   Case $j = r$**:** Compute $c_r^{(v)} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c_r^{(v-1)})$ for $v \in \{j_b, \ldots, n-1\}$ where $c_r^{(j_b - 1)} = w \leftarrow\!\!\$\ \mathcal{M}_{2,j_b}$. Send the challenge $(\mathbb{C}^{\mathsf{in}}_{c_r^{(n-1)}, \mathsf{sk}_r, r}, (\mathsf{sk}_r, r))$ to the challenger and receive the answer $\widetilde{\mathbb{C}}^*$. Set $c_r = (\widetilde{\mathbb{C}}^{\mathsf{in}}_r, \widetilde{\mathbb{C}}^{\mathsf{out}}_r)$ where $\widetilde{\mathbb{C}}^{\mathsf{in}}_r = \widetilde{\mathbb{C}}^*$, $y^{\mathsf{out}}_r \leftarrow\!\!\$\ \{0,1\}^{s_4(\lambda)}$, and $\widetilde{\mathbb{C}}^{\mathsf{out}}_r \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c_r^{(n-1)}, \mathsf{sk}_r, r}, y^{\mathsf{out}}_r, m^b_r)$.

   Case $j \in \{r+1 \mod n, \ldots, j_b - 1 \mod n\}$**:** Compute $c_j \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{ek}_j, x^b_j, m^b_j)$.
5. Send the challenge ciphertexts $(c_0, \ldots, c_{n-1})$ 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 $\mathbf{H}^{b,0,0}_{7+i}(\lambda)$. On the other hand, if $d = 1$, A simulates $\mathbf{H}^{b,1,0}_{7+i}(\lambda)$. Hence, A has the same advantage of D. This concludes the proof. □

Claim 36. $\mathbf{H}^{b,1,0}_{7+i}(\lambda) \approx_c \mathbf{H}^{b,1,1}_{7+i}(\lambda)$ *for* $i \in [n-1]$.

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

Claim 37. $\mathbf{H}^{1-b,1,1}_{7+n-1}(\lambda) \approx_c \mathbf{H}^{b,1,1}_{7+n-1}(\lambda)$.

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

By combining Claims 28 to 37 and conditioned to the event $\mathbf{Validity}_{2,j_0,j_1}$, we conclude that

$$\begin{aligned}\mathbf{H}^b_0 \approx_c \mathbf{H}^b_1 \equiv \mathbf{H}^{b,0}_2 \approx_c \ldots \approx_c \mathbf{H}^{b,q}_2 \approx_c \mathbf{H}^b_3 \equiv \mathbf{H}^b_4 \approx_c \ldots \approx_c \mathbf{H}^b_4 \equiv \\ \mathbf{H}^{b,0}_5 \approx_c \ldots \approx_c \mathbf{H}^{b,q}_5 \equiv \mathbf{H}^{b,0}_6 \approx_c \ldots \approx_c \mathbf{H}^{b,q}_6 \equiv \mathbf{H}^{b,1,1}_7 \\ \approx_c \mathbf{H}^{b,0,0}_8 \approx_c \ldots \approx_c \mathbf{H}^{b,1,1}_{7+n-1} \equiv \mathbf{H}^{1-b,1,1}_{7+n-1}.\end{aligned}$$

This concludes the proof. □

Lemma 10. *Let* $j_0 \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$. *If* PE *is CPA secure without collusion (Definition 8),* PKE_{2,j_0} *is CPA secure (Definition 6),* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then*

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

Proof. Without loss of generality, let $q = |\mathcal{Q}_{j_0}| \in \mathsf{poly}(\lambda)$ where $j_0 \notin \mathcal{Q}_{\mathsf{Corr}}$. Consider the hybrid experiments of Lemma 4 and Lemma 9. Formally,

- Let $\mathbf{H}^{1,i}_0(\lambda), \mathbf{H}^{1,i}_1(\lambda), \mathbf{H}^{1,i}_2(\lambda)$, for $i \in \{-1, 0, \ldots, n-1\}$, be the hybrid of Lemma 8 (for the challenge bit $b = 1$) except that are conditioned to the event $\mathbf{Validity}_{3,j_0}$ (instead of $\mathbf{Validity}_1$)
- Let $\mathbf{H}^0_0(\lambda), \mathbf{H}^0_1(\lambda), \mathbf{H}^{0,t}_2(\lambda), \mathbf{H}^0_3(\lambda), \mathbf{H}^0_4(\lambda), \mathbf{H}^{0,t}_5(\lambda), \mathbf{H}^{0,t}_6(\lambda), \mathbf{H}^{0,1,1}_7(\lambda), \mathbf{H}^{0,k_1,k_2}_{7+j}(\lambda)$, for $i \in [n-1], t \in [0,q], (k_1,k_2) \times \{0,1\}^2$, be the hybrids of Lemma 9 (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:

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

$\mathbf{H}^{0,i}_{7+n}$**:** Same as $\mathbf{H}^{0,i-1}_{7+n}$ 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 $i' \leq i$, the challenger returns $c_{j_0} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_0}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_0})$ where $\widetilde{\mathbb{C}}^{\mathsf{out}}_v \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c^{(n-1)}_{j_0}, \mathsf{sk}_{j_0}, j_0}, y^{\mathsf{out}}_{j_0}, m)$ where $y^{\mathsf{out}}_{j_0} \leftarrow\!\!\$\ \{0,1\}^{s_4(\lambda)}$. Otherwise, on input the i'-th query (x, m) such that $i' > i$, the challenger answers as usual, i.e., as defined in $\mathbf{H}^{0,0}_{7+n}$.

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

$\mathbf{H}^{0,i,}_{7+n+1}$**:** Same as $\mathbf{H}^{0,i-1}_{7+n+1}$ 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 $i' \leq i$, the challenger returns $c_{j_0} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_0}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_0})$ where $\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_0} \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c^{(n-1)}_{j_0},\mathsf{sk}_{j_0},j_0}, y^{\mathsf{in}}_{j_0}, (\mathsf{sk}_{j_0}, j_0))$ where $y^{\mathsf{in}}_{j_0} \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)}$. Otherwise, on input the i'-th query (x,m) such that $i' > i$, the challenger answers as usual, i.e., as defined in $\mathbf{H}^{0,0}_{7+n+1}$.

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

$\mathbf{H}^{0,i}_{7+n+2}$**:** Same as $\mathbf{H}^{0,i-1}_{7+n+2}$ 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 $i' \leq i$, the challenger samples $(y^{\mathsf{in}}_{j_0}, y^{\mathsf{out}}_{j_0}) \leftarrow\!\!\$\ \{0,1\}^{s_3(\lambda)+s_4(\lambda)}$ and computes $c^{(-1)}_{j_0} \leftarrow\!\!\$\ \mathsf{Enc}_1(\mathsf{mpk}, (x_0,\ldots, x_{n-1}), y^{\mathsf{in}}_{j_0}||y^{\mathsf{out}}_{j_0})$ where $x_{j_0} = x$, and $x_j = x^\star_j$ for $j \in \{0,\ldots,n-1\} \setminus \{j_0\}$. Finally, the challenger returns $c_{j_0} = (\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_0}, \widetilde{\mathbb{C}}^{\mathsf{out}}_{j_0})$ where $c^{(v)}_{j_0} \leftarrow\!\!\$\ \mathsf{Enc}_{2,v}(\mathsf{pk}_v, c^{(v-1)}_{j_0})$ for $v \in \{0,\ldots,n-1\}$, $\widetilde{\mathbb{C}}^{\mathsf{in}}_{j_0} \leftarrow\!\!\$\ \leftarrow\!\!\$\ \mathsf{Obf}_3(1^\lambda, \mathbb{C}^{\mathsf{in}}_{c^{(n-1)}_{j_0},\mathsf{sk}_{j_0},j_0}, y^{\mathsf{in}}_{j_0}, (\mathsf{sk}_{j_0}, j_0))$, and $\widetilde{\mathbb{C}}^{\mathsf{out}}_v \leftarrow\!\!\$\ \mathsf{Obf}_4(1^\lambda, \mathbb{C}^{\mathsf{out}}_{c^{(n-1)}_{j_0},\mathsf{sk}_{j_0},j_0}, y^{\mathsf{out}}_{j_0}, m)$. Otherwise, on input the i'-th query (x,m) such that $i' > i$, the challenger answers as usual, i.e., as defined in $\mathbf{H}^{0,0}_{7+n+2}$.

Claim 38. $\mathbf{H}^0_0(\lambda) \approx_c \mathbf{H}^{0,1,1}_{7+n-1}(\lambda)$.

Proof. The proof of Claim 38 is identical to that of Lemma 5 where the challenge bit is $b = 0$. □

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

Proof. Claim 39 follows by leveraging a similar argument to that of Claim 33. □

Claim 40. $\mathbf{H}^{0,i-1}_{7+n+1}(\lambda) \approx_c \mathbf{H}^{0,i}_{7+n+1}(\lambda)$ $i \in [q]$.

Proof. Claim 40 follows by leveraging a similar argument to that of Claim 32. □

Claim 41. $\mathbf{H}^{0,i-1}_{7+n+2}(\lambda) \approx_c \mathbf{H}^{0,i-1}_{7+n+2}(\lambda)$ *for* $i \in [q]$.

Proof. Claim 41 follows by leveraging a similar argument to that of Claim 29. □

Claim 42. $\mathbf{H}^{1,-1}_0(\lambda) \approx_c \mathbf{H}^{1,q}_2(\lambda)$.

Proof. The proof of Claim 42 is identical to that of Lemma 4 where the challenge bit is $b = 1$. □

Claim 43. $\mathbf{H}^{0,q}_{7+n+2}(\lambda) \equiv \mathbf{H}^{1,q}_2(\lambda)$.

Proof. Claim 43 follows by observing that experiments $\mathbf{H}^{0,q}_{7+n+2}(\lambda)$ and $\mathbf{H}^{1,q}_2(\lambda)$ are identical (and does not depend on the bit b). □

By combining Claims 38 to 43 and conditioned to the event $\mathbf{Validity}_{3,j_0,}$, we conclude that

$$\begin{aligned}\mathbf{H}^0_0 \approx_c \mathbf{H}^{0,1,1}_{7+n-1} \equiv \mathbf{H}^{b,0}_{7+n} \approx_c \ldots \approx_c \mathbf{H}^{b,q}_{7+n} \equiv \mathbf{H}^{b,0}_{7+n+1} \approx_c \ldots \approx_c \mathbf{H}^{b,q}_{7+n+1}\\ \equiv \mathbf{H}^{b,0}_{7+n+2} \approx_c \ldots \approx_c \mathbf{H}^{b,q}_{7+n+2} \equiv \mathbf{H}^{1,q}_2 \approx_c \mathbf{H}^{1,-1}_0.\end{aligned}$$

This concludes the proof. □

Lemma 11. *Let $j_1 \in [0, n-1] \setminus \mathcal{Q}_{\mathsf{Corr}}$. If* PE *is CPA secure without collusion (Definition 8),* PKE_{2,j_1} *is CPA secure (Definition 6),* LOBF_3 *and* LOBF_4 *are secure (Definition 2), then*

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

Proof. Lemma 11 follows by using a symmetrical argument to that of Lemma 10. □

By combining Lemmas 8 to 11 we conclude that Π is CPA secure in the $(n-1)$-corruptions setting without collusion.