CCA Secure A Posteriori Openable Encryption in the Standard Model

Xavier Bultel

CCA Secure *A Posteriori* Openable Encryption in the Standard Model*

Xavier Bultel

INSA Centre Val de Loire, Laboratoire dinformatique fondamentale dOrléans, France
xavier.bultel@insa-cvl.fr

Abstract. *A Posteriori* Openable Public Key Encryptions (APOPKE) allow any user to generate a constant-size key that decrypts the messages they have sent over a chosen period of time. As an important feature, the period can be dynamically chosen after the messages have been sent. This primitive was introduced in 2016 by Bultel and Lafourcade. They also defined the Chosen-Plaintext Attack (CPA) security for APOPKE, and designed a scheme called GAPO, which is CPA secure in the random oracle model. In this paper, we formalize the Chosen-Ciphertext Attack (CCA) security for APOPKE, then we design a scheme called CHAPO (for *CHosen-ciphetext attack resistant A Posteriori Openable encryption*), and we prove its CCA security in the standard model. CHAPO is approximately twice as efficient as GAPO and is more generic. We also give news applications, and discuss the practical impact of its CCA security.

1 Introduction

Over the past decade, the development of smartphones has made instant messaging increasingly accessible and popular with the general public. The security of such applications has become a crucial issue since Edward Snowden's revelations in 2013, which made people aware of the importance of protecting their personal data on the Internet. Today, there are many encrypted messaging protocols that allow anyone to protect their private conversations from hackers and mass surveillance, such as Signal, Telegram, Whatsapp, Wire, Keybase, or Line. However, because they are within everyone's reach, these tools are also used by criminals. The example of the terrorist group Daesh, which uses Telegram to communicate, is a good illustration of this. For that reason, encrypting the communications can be seen as a suspicious act, and can harm a user during a legal case, especially if the judicial authority requests the assistance of the user's Internet service provider to use lawful interception: if a user encrypts his communications, then lawful interception will only intercept unintelligible data. In such a scenario, the authorities requisition the smartphone and the user has to reveal his access codes and secret keys so that the investigators exculpate him. However, the investigators get unlimited access to the user's data, and can get to know private data that does not concern the investigation.

A Posteriori Openable Public Key Encryption (APOPKE), introduced by Bultel and Lafourcade in [5], offers a practical solution to this problem. In addition to the standard features of public key encryptions, an APOPKE scheme implements a mechanism that allows a user to generate a special key called *interval key* from any pair of indexes i and j, which allows a user called *the opener* to decrypt the messages sent by the user (potentially from several different devices) from the i^{th} to the j^{th}. The interval can be chosen *a posteriori*, *i.e.* after the user has sent the messages. In addition, since the resources of user's device(s) are potentially limited, the interval key must be generated in constant time using a single secret key, and must be of constant size: this prevents the user from storing too much data, from having to perform a large number of operations on data spread over several devices, or from having to send very large data to the opener growing with the size of the interval. Moreover, the complexity of the encryption algorithm and the size of the ciphertexts should not depend on the size of the interval for similar efficiency reasons. Note that the interval key allows to decrypt any message sent between the indexes i and j, regardless of the public key of the recipient of the message. Let us consider that an instant messaging uses an APOKPE scheme, and that each encrypted message is dated and stored on the server of the application. If a user wants to reveal his conversations over a given time period to a judge during an investigation, then he can generate an interval key with the indexes of the first and

* This work has been accepted for publication in the proceedings of the **CT-RSA 2022** conference.

last messages sent during this period. The judge can then decrypt and view the messages stored on the server for the time period.

In [5], the authors define a security model for the *Chosen-Plaintext Attack* (CPA) security and for the *integrity* of APOPKE. The integrity property ensures that the messages decrypted by the opener with the interval keys are the same as the one decrypted by the recipients of the messages. The authors also design a scheme that is proven CPA secure and has the integrity property in the so-called Random Oracle Model (ROM). They leave as an open problem the design of a scheme secure against the Chosen-Ciphertext Attacks (CCA), and the design of a scheme proven secure in the standard model. In this paper, we solve these two problems. We also show that CPA security is insufficient in practice for the application of this primitive in secure messaging, but also in the applications proposed in [5].

Our contribution. In this paper, we extend the security model proposed by Bultel and Lafourcade in order to capture the CCA security of the APOPKE. We instantiate our model with a new scheme called CHAPO, then we present some applications for the (CCA secure) APOPKE on encrypted messaging and voice calls. As in [5], we distinguish two CCA security notions: *Indistinguishability Against Chosen Ciphertext Attack* security (IND-CCA), and *Indistinguishability Against Chosen Set of Ciphertext Attack* (IND-CSCA) security.

IND-CCA: this property is an extension of the IND-CCA property for PKE. The adversary is a malicious opener who tries to guess which message has been encrypted by the challenger between two chosen messages. This adversary has access to oracles that encrypt, decrypt, and generate an interval key using the secrets of some honest users. In order to avoid trivial attacks, the challenge ciphertext cannot be decrypted by the oracles, and cannot be decrypted by the queried interval key.

IND-CSCA: since the opening mechanism of an APOPKE can be viewed as an alternate decryption algorithm for the opener (where the ciphertext is the association of the interval key with the set of the ciphertexts in the interval), our CCA model must consider the attacks where the adversary has access to an oracle that decrypts a set of ciphertexts using the opening algorithm. The IND-CSCA property considers a collusion of dishonest users who receives the interval keys of an honest opener. These users try to learn some information about the messages encrypted for the honest users, and can query some oracles that encrypt, decrypt, generate interval keys, and open intervals, using the secrets of the honest users and the honest opener.

Note that the two properties IND-CCA and IND-CSCA are complementary. They capture two different adversary models, *i.e.* a collusion between the opener and some users trying to attack a ciphertext outside the interval, and a collusion of users attacking the ciphertexts in the interval without the help of the opener. Neither of the two involves the other. Our second contribution is a new APOPKE scheme called CHAPO, that have the following features:

Security: We prove that CHAPO is IND-CCA and IND-CSCA secure in the standard model. Moreover we prove its integrity according to the definition given in [5].

Genericity: Our scheme is based on standard cryptographic tools: it requires a Pseudorandom Fonction (PRF), a Message Authentication Code scheme (MAC), a Symmetric Key Encryption scheme (SKE) and a Public Key Encryption scheme (PKE). We stress that unlike GAPO, there is no restriction in the choice of the PKE, making CHAPO more generic. For instance, CHAPO can be instantiated with the CPA variant of the McEliece cryptosystem [20] (for a CPA secure instantiation), which is not possible for GAPO since it is not *random coin decryptable* (according to a definition given in [5]), *i.e.* the random coin used by the encryption algorithm cannot be used as an alternative secret key to decrypt the corresponding ciphertext.

Efficiency. The dominating operations in CHAPO are the encryptions and decryptions of the PKE. The encryption/decryption of a message requires one PKE encryption/decryption, and the opening algorithm on a set of n ciphertexts requires n PKE encryptions. In comparison, the GAPO encryption/decryption of a message requires 2 PKE encryptions/decryptions, and the opening algorithm of GAPO on a set of n ciphertexts requires $2 \cdot n$ PKE encryptions and $2 \cdot n$ PKE decryptions by random-coin. Finally, instantiated with a CPA secure PKE, CHAPO is approximately twice as efficient as GAPO for any instantiation that yields the same level of security (CPA).

As in [5], our model considers only a single dishonest opener (or several openers who do not collude) who receives only one interval key. Actually, we relax this constraint a bit compared to [5] since we allow the opener to receive keys for several intervals as long as the index of the challenge is not between the

lowest and the greatest index of the queried intervals. We let the design of a secure scheme that allows multiple disjoint (constant-size) interval keys as an open problem.

Related works. Some encryption primitives have time-dependent decryption properties, and are therefore related to our work. Time-Release Encryption (TRE) [6, 19] allows to produce ciphertexts that cannot be decrypted before a given date. Some schemes implement a pre-open mechanism that allows to decrypt the ciphertexts before the date [14]. In this case, both decryption methods must return the same message; this property, called *binding property*, is similar to the APOPKE integrity property. Time-Specific Encryption (TSE) [21] extends the TRE concept: the encryption algorithm produces ciphertexts that can be decrypted in a pre-defined time interval only. Even if this line of research is close to ours, the goals of TSE differ significantly from APOPKE: in TSE, the time is bounded by a max value T, and the time interval is chosen *a priori* in $[0, T-1]$, *i.e.* before than the message is encrypted. TSE constructions are less generic than CHAPO [15, 16], and there is no scheme that ensures that both ciphertexts and keys are of constant size [15]. Key-Insulated Encryption (KIE) [8, 25] deals with the problem of storing secret keys on insecure device. The keys are refreshed periodically by a secure server using a master secret key. The encryption algorithm encrypts the messages from a time period tag and a public key that remains unchanged over time. In [8] the authors show that KIE can be used to delegate the decryption on a time interval by sending the keys that match the interval. However, the periods are fixed, which avoids the fine-grained management of the opener decryption capabilities if the interval does not perfectly match the periods, and the opener decryption key grows linearly with the number of periods. Moreover, this solution allows the opener to decrypt messages received by the user, whereas APOPKE allows the opener to decrypt messages sent by the user (potentially with different public keys), which constitutes an important conceptual break between these two primitives. Delegatable pseudorandom functions [17] allow to delegate the computation of a pseudorandom function over an interval of inputs. By using such a function to generate the encryption keys of successive ciphertexts, a user can delegate the generation of decryption keys over a given interval, and thus obtain features similar to APOPKE. However, using [17] the total number of ciphertexts is bounded and the interval key is logarithmic in this bound, whereas an APOPKE must allow the generation of an unlimited number of ciphertexts while ensuring an interval key of constant size.

On the application side, Key-Escrow (KE) and Key-Recovery (KR) [13, 18, 23, 24] schemes deal with the capability of an authority to decrypt the messages exchanged between two users by recovering their secret key. Some works deal with KR mechanisms for lawful interception [1, 13, 24], where a set of authorities has to collude to open an encrypted phone conversation between two users. As in APOPKE, the security model ensures that the recovered conversation is the same as the exchanged one. However, the motivations of such schemes differ from ours since the authorities recover the keys one by one, contrary to APOPKE where a single key makes it possible to open all the messages between two dates. In this paper, we discuss the advantage of using our solution in the voice call lawful interception context.

Finally, to the best of our knowledge, with the exception of [5], there are no encryption schemes with features similar to CHAPO. Note that such a scheme could be obtained with generic primitives such as functional encryption [10], by considering the set of ciphertexts as the blocks of a single large ciphertext, and by considering a function that returns the plaintexts of an interval on the blocks. However, such a solution would be inefficient and would limit the number of possible ciphertexts. We could also use attribute-based encryption [12], and give the judge a key to decrypt on the attributes used during the time interval, but to the best of our knowledge, no attribute-based encryption scheme offers both constant-size ciphertexts and keys, and the limit on the number of attributes would limit the number of possible messages.

2 Cryptographic Tools

Notation. In this paper, $y \leftarrow \mathsf{Alg}(x)$ denotes that the output of the deterministic algorithm Alg running on the input x is assigned to y. Moreover, $\mathsf{Alg}(x; r)$ denotes that the probabilistic algorithm Alg uses the random coin r to generate its randomness. If the context is clear, we omit to mention the random coin in the description of some algorithms, especially for adversaries (denoted $\mathcal{A}$) and key generators (denoted Gen). $x \leftarrow y$ denotes the assignment of the value of y to the variable x, and $x \xleftarrow{\$} X$ denotes the random sample of x in the uniform distribution on X. We assume that any variable has the value

$\perp$ by default, which means that it is not defined, or not instantiated. For any security property denoted Prop defined in this paper, any cryptographic scheme P, and any Probabilistic Polynomial Time (PPT) algorithm $\mathcal{A}$, we define an avantage denoted $\mathbf{Adv}^{\mathsf{Prop}}_{P,\mathcal{A}}(\lambda)$, where λ is a security parameter. We say that P is Prop-secure when the following function, called the Prop-advantage of P, is negligible: $\mathbf{Adv}^{\mathsf{Prop}}_{P}(\lambda) = \max_{\mathcal{A}\in \text{POLY}(\lambda)} \left\{ \mathbf{Adv}^{\mathsf{Prop}}_{P,\mathcal{A}}(\lambda) \right\}$.

Definition 1 (PRF [9]). *Let λ be a security parameter, $\mathcal{K}_\lambda$ be a set, and (α, β) be two integers. A* Pseudorandom Function *(PRF) is a function $\mathsf{PRF} : \mathcal{K}_\lambda \times \{0,1\}^\alpha \to \{0,1\}^\beta$.*
For any PPT algorithm $\mathcal{A}$, and for $\mathcal{F} = \{f : \{0,1\}^\alpha \to \{0,1\}^\beta\}$, the advantage of $\mathcal{A}$ on the pseudorandomness of PRF is defined by the function: $\mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF},\mathcal{A}}(\lambda) =$

$$\left| \mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F} : 1 \leftarrow \mathcal{A}^{f(\cdot)}(\lambda) \right] - \mathsf{Pr}\left[\begin{array}{r} \mathsf{rk} \xleftarrow{\$} \mathcal{K}_\lambda; \\ f(.) \leftarrow \mathsf{PRF}(\mathsf{rk}, \cdot); \end{array} : 1 \leftarrow \mathcal{A}^{f(\cdot)}(\lambda) \right] \right|$$

Definition 2 (MAC [3]). *Let λ be a security parameter. A* Message Authentication Code scheme *(MAC) is defined by a tuple $(\mathcal{K}_\lambda, \mathsf{Mac}, \mathsf{Ver})$ such that $\mathcal{K}_\lambda$ is a set, and the algorithms Mac and Ver are defined by:*
$\underline{\mathsf{Mac}(\mathsf{mk}, m)}$**:** *On input a key $\mathsf{mk} \in \mathcal{K}_\lambda$, and a message $m \in \{0,1\}^*$, return a tag t. This algorithm is deterministic.*
$\underline{\mathsf{Ver}(\mathsf{mk}, t, m)}$**:** *On input a key $\mathsf{mk} \in \mathcal{K}_\lambda$, a tag t, and a message $m \in \{0,1\}^*$. Return a bit b.*
Let P be a MAC scheme, and let $\mathcal{A}$ be a PPT algorithm. The Existential UnForgeability against Chosen Message Attack *($\mathsf{EUF\text{-}CMA}$) security experiment for P is defined by:*

$\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{P,\mathcal{A}}(\lambda)$*:*
$\mathcal{S} \leftarrow \emptyset$*;*
$\mathsf{mk}_* \xleftarrow{\$} \mathcal{K}_\lambda$*;*
$(t_*, m_*) \leftarrow \mathcal{A}^{\mathsf{Mac}(\mathsf{mk}_*,\cdot),\mathsf{Ver}(\mathsf{mk}_*,\cdot,\cdot)}(\lambda)$*;*
If $1 = \mathsf{Ver}(\mathsf{mk}_*, t_*, m_*)$ *and* $(t_*, m_*) \notin \mathcal{S}$*,*
then return 1*, else* 0*;*

where the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$ takes a message m as input, runs $t \leftarrow \mathsf{Mac}(\mathsf{mk}_*, m)$, updates $\mathcal{S} \leftarrow \mathcal{S} \cup \{(t, m)\}$ and returns t, and the oracle $\mathsf{Ver}(\mathsf{mk}_*, \cdot, \cdot)$ takes (t, m) as input, runs $b \leftarrow \mathsf{Ver}(\mathsf{mk}_*, t, m)$ and returns b. The $\mathsf{EUF\text{-}CMA}$-advantage of $\mathcal{A}$ is defined by $\mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{P,\mathcal{A}}(\lambda) = \mathsf{Pr}[\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{P,\mathcal{A}}(\lambda) = 1]$.

Definition 3 (SKE [4]). *Let λ be a security parameter. A* Symmetric Key Encrytion scheme *(SKE) is defined by a tuple $(\mathcal{K}_\lambda, \mathcal{R}_\lambda, \mathsf{SEnc}, \mathsf{SDec})$ such that $\mathcal{K}_\lambda$, and $\mathcal{R}_\lambda$ are two sets, and the algorithms SEnc and SDec are defined by:*
$\underline{\mathsf{PEnc}(\mathsf{ek}, m; r)}$**:** *On input a key $\mathsf{ek} \in \mathcal{K}_\lambda$, a message $m \in \{0,1\}^*$, and a random coin $r \in \mathcal{R}_\lambda$, return a ciphertext c.*
$\underline{\mathsf{PDec}(\mathsf{ek}, c)}$**:** *On input a key $\mathsf{ek} \in \mathcal{K}_\lambda$, and a ciphertext c, return a message m.*
Let P be a SKE, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a pair of PPT algorithms. The Indistinguishability against Chosen Ciphertexts Attacks security *($\mathsf{IND\text{-}CCA}$) security experiment for P is defined by:*

$\mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)$*:*
$\mathsf{ek}_* \xleftarrow{\$} \mathcal{K}_\lambda$*;*
$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathsf{SEnc}(\mathsf{ek}_*,\cdot),\mathsf{SDec}(\mathsf{ek}_*,\cdot)}(\lambda)$*;*
If $|m_0| \neq |m_1|$*, then return a random bit;*
$r \xleftarrow{\$} \mathcal{R}_\lambda$*;* $c_* \leftarrow \mathsf{Enc}(\mathsf{ek}_*, m_b; r)$*;*
$b_* \leftarrow \mathcal{A}_2^{\mathsf{SEnc}(\mathsf{ek}_*,\cdot),\mathsf{SDec}(\mathsf{ek}_*,\cdot)}(\lambda, \mathsf{st}, c_*)$*;*
If $b = b_*$*, then return* 1*, else* 0*;*

where the oracle $\mathsf{SEnc}(\mathsf{ek}_*, \cdot)$ takes a message m as input, picks a random coin $r \xleftarrow{\$} \mathcal{R}_\lambda$, and returns $\mathsf{PEnc}(\mathsf{ek}_*, m; r)$, and the oracle $\mathsf{SDec}(\mathsf{ek}_*, \cdot)$ takes a ciphertext c as input, and returns $\mathsf{SDec}(\mathsf{ek}_*, c)$, except if $c = c_*$ during the second phase. The $\mathsf{IND\text{-}CCA}$-advantage of $\mathcal{A}$ is defined by $\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{P,\mathcal{A}}(\lambda) = |\mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)] - 1/2|$.

Definition 4 (PKE [11]). *Let λ be a security parameter. A* Public Key Encryption *scheme (PKE) is defined by a tuple $P = (\mathcal{R}_\lambda, \mathcal{M}_\lambda, \mathsf{PGen}, \mathsf{PEnc}, \mathsf{PDec})$ such that $\mathcal{R}_\lambda$ is a set, and the algorithms PGen, PEnc, and PDec are defined by:*
$\underline{\mathsf{PGen}(1^\lambda)}$**:** *Return a pair of public/secret keys $(\mathsf{pk}, \mathsf{sk})$.*

$\underline{\mathsf{PEnc}(\mathsf{pk}, m; r)}$**:** *On input a public key* pk*, a message* $m \in \mathcal{M}_\lambda$*, and a random coin* $r \in \mathcal{R}_\lambda$*, return a ciphertext* c*.*

$\underline{\mathsf{PDec}(\mathsf{sk}, c)}$**:** *On input a secret key* sk *and a ciphertext* c*, return a message* m*.*

P is said to be correct if for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PGen}(1^\lambda)$*, any* $m \in \mathcal{M}_\lambda$ *and any* $r \in \mathcal{R}_\lambda$ *it holds that* $m = \mathsf{PDec}(\mathsf{sk}, \mathsf{PEnc}(\mathsf{pk}, m; r))$*.*

Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *be a pair of PPT algorithms. The* Indistinguishability against Chosen-Ciphertext attack *(*IND-CCA*) security experiment for P is defined by:*

$\mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)$*:*

$(\mathsf{pk}_*, \mathsf{sk}_*) \leftarrow \mathsf{PGen}(1^\lambda)$*;*

$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathsf{PDec}(\mathsf{sk}_*, \cdot)}(\lambda, \mathsf{pk}_*)$*;*

$r \xleftarrow{\$} \mathcal{R}_\lambda; c_* \leftarrow \mathsf{PEnc}(\mathsf{pk}_*, m_b; r)$*;*

$b_* \leftarrow \mathcal{A}_2^{\mathsf{PDec}(\mathsf{sk}_*, \cdot)}(\lambda, \mathsf{pk}_*, \mathsf{st}, c_*)$*;*

If $b = b_*$*, then return* 1*, else* 0*;*

where the oracle $\mathsf{PDec}(\mathsf{sk}_*, \cdot)$ *takes a ciphertext* c *as input, and returns* $\mathsf{PDec}(\mathsf{sk}_*, c)$*, except if* $c = c_*$ *during the second phase. The* IND-CCA*-advantage of* $\mathcal{A}$ *is defined by* $\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{P,\mathcal{A}}(\lambda) = |\mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)] - 1/2|$*.*

Finally, we recall the notion of *Verifiable Key* defined in [5].

Definition 5 (Verifiable key generator). *A public/private keys pair generation algorithm* PGen *is verifiable if there exists a deterministic polynomial-time algorithm* KVer *such that, for any security parameter* λ*, it holds that:*
$\mathsf{KVer}(\mathsf{pk}_*, \mathsf{sk}_*) = 1 \Leftrightarrow (\mathsf{pk}_*, \mathsf{sk}_*) \in \{(\mathsf{pk}, \mathsf{sk}) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PGen}(1^\lambda)\}$.

We show a simple trick to turn any key generator PGen into a verifiable key generator PGen'. In what follows we explicit the use of the random coin (denoted r) by the key generator: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PGen}(1^\lambda; r)$. We define a new key generator PGen' that runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PGen}(1^\lambda; r)$ and returns $(\mathsf{pk}', \mathsf{sk}') = (\mathsf{pk}, (\mathsf{sk}, r))$. The generator PGen' can be used to replace PGen because it outputs the same public key as the one returned by PGen, and it returns a secret key sk' that contains the secret key sk outputted by PGen. Moreover, PGen' is verifiable because $\mathsf{KVer}(\mathsf{pk}_*, \mathsf{sk}_*)$ can be instantiated by the following algorithm: parse sk_* as (sk, r), run $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{PGen}(1^\lambda; r)$, if $\mathsf{pk}_* = \mathsf{pk}'$ and $\mathsf{sk}_* = (\mathsf{sk}', r)$, then return 1, else return 0. Thus, in this paper, we will implicitly consider that any key generator is verifiable.

3 Formal Definitions

An APOPKE is defined by a *public/private key pair generator*, an *encryption key generator* that generates an encryption key for each user, an *encryption algorithm* that encrypts a message from its index, the user encryption key, and the recipient public key, a *decryption algorithm* that decrypts a ciphertext from the secret key of the recipient, an *extractor algorithm* that generates an interval key for an opener from two indexes (i, j), the user encryption key and the opener public key, and an *opening algorithm* that allows the opener to decrypt each message labelled by an index in $[\![i, j]\!]$ using his interval key and his secret key.

Definition 6 (APOPKE). *Let* λ *be a security parameter. An APOPKE scheme is defined by a tuple* $(\mathcal{R}_\lambda, \mathcal{M}_\lambda, \mathsf{Gen}, \mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext}, \mathsf{Open})$ *such that* $\mathcal{R}_\lambda$ *and* $\mathcal{M}_\lambda$ *are two sets, and are defined by:*

$\underline{\mathsf{Gen}(1^\lambda)}$**:** *Return a pair of public/secret keys* $(\mathsf{pk}, \mathsf{sk})$*. The same algorithm is used to generate the keys of the users and the opener. In what follows, we will use the notation* $(\mathsf{pko}, \mathsf{sko})$ *to designate the opener keys.*

$\mathsf{EGen}(1^\lambda)$**:** *Return an encryption secret key* k*.*

$\underline{\mathsf{Enc}(\mathsf{pk}_l, \mathsf{k}, m_l, l; r_l)}$**:** *On input a receiver public key* pk_l*, a sender secret encrytion key* k*, a message* $m_l \in \mathcal{M}_\lambda$*, an index* $l \in \mathcal{N}$*, and a random coin* $r_l \in \mathcal{R}_\lambda$*, return a ciphertext* c_l*.*

$\mathsf{Dec}(\mathsf{sk}_l, c_l)$**:** *On input a receiver secret key* sk_l *and a ciphertext* c_l*, return a message* m_l*.*

$\underline{\mathsf{Ext}(\mathsf{pko}, \mathsf{k}, i, j; r)}$**:** *On input an opener public key* pko*, a sender secret encrytion key* k*, two indexes* i *and* j*, and a random coin* $r \in \mathcal{R}_\lambda$*. Return an interval key* $\mathsf{ik}_{i \to j}$*.*

$\underline{\mathsf{Open}(\mathsf{sko}, i, j, \mathsf{ik}_{i \to j}, (c_l, \mathsf{pk}_l)_{i \le l \le j})}$**:** *On input an opener secret key* sko*, two indexes* i *and* j*, an interval key* $\mathsf{ik}_{i \to j}$*, and a vector of ciphertexts couped with their respective public keys* $(c_l, \mathsf{pk}_l)_{i \le l \le j}$*. Return messages* $(m_l)_{i \le l \le j}$*.*

The IND-CCA experiment features an adversary who knows the public key of an honest user, chooses two messages, receives the encryption of one of them as challenge, and tries to guess which one has been encrypted with that public key. The adversary has access to an encryption oracle that encrypts chosen messages using an index n incremented by the experiment after each encryption and the (secret) encryption key of the honest user, as well as a decryption oracle that decrypts any chosen ciphertext except the challenge. At the beginning of the experiment, n is initialized to 1. Note that the experiment simulates an honest user encrypting several successive messages. Allowing forks in the ciphertext sequence would introduce trivial and unrealistic attacks to deal with (in practice the attacked user would have no interest in using non-successive indices) artificially complicating the model and the proofs. In addition, the adversary has access to an extraction oracle that generates interval keys for chosen intervals. However, the adversary cannot query the oracle for intervals (i_k, j_k) such that the index of the challenge is between $\min_k(i_k)$ and $\max_k(j_k)$.

Definition 7 (IND-CCA experiment). *Let $P = (\mathcal{R}_\lambda, \mathcal{M}_\lambda, \mathsf{Gen}, \mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext}, \mathsf{Open})$ be an* APOPKE*, let λ be a security parameter, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a pair of PPT algorithms. The* Indistinguishability against Chosen Ciphertext Attack *(*IND-CCA*) security experiment for P is defined by:*

$\mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)$*:*

$n \leftarrow 1$*;* $(i_*, j_*) \leftarrow (0,0)$*;*

$(\mathsf{pk}_*, \mathsf{sk}_*) \leftarrow \mathsf{Gen}(1^\lambda)$*;*

$\mathsf{k}_* \leftarrow \mathsf{EGen}(1^\lambda)$*;*

$\mathcal{O} \leftarrow \left\{ \begin{array}{l} \mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n); \mathsf{Dec}(\mathsf{sk}_*, \cdot); \\ \mathsf{Ext}(\cdot, \mathsf{k}_*, \cdot, \cdot); \end{array} \right\}$*;*

$(m_{(*,0)}, m_{(*,1)}, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\lambda, \mathsf{pk}_*)$*;*

$l_* \leftarrow n$*;* $r_* \xleftarrow{\$} \mathcal{R}_\lambda$*;*

$c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,b)}, l_*; r_*)$*;*

$n \leftarrow n + 1$*;*

$b_* \leftarrow \mathcal{A}_2^{\mathcal{O}}(\lambda, \mathsf{pk}_*, \mathsf{st}, c_{l_*})$*;*

If $l_* \in [\![i_*, j_*]\!]$, then return a random bit;

If $b = b_*$, then return 1, else 0;

where the oracles are defined by:

$\underline{\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)}$**:** *On input (pk_n, m_n), pick $r_n \xleftarrow{\$} \mathcal{R}_\lambda$, run $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ and increment $n \leftarrow (n+1)$, then return c_n.*

$\underline{\mathsf{Dec}(\mathsf{sk}_*, \cdot)}$**:** *On input c, during the second phase if $c_{l_*} = c$, then return $\perp$. Finally, run $m \leftarrow \mathsf{Dec}(\mathsf{sk}_*, c)$ and return m.*

$\underline{\mathsf{Ext}(\cdot, \mathsf{k}_*, \cdot, \cdot)}$**:** *On input (pko, i, j), if $i \leq 0$ or $j \leq 0$ or $i \geq j$ or $j \geq n$, then return $\perp$. if $i_* = j_* = 0$, then set $(i_*, j_*) \leftarrow (i, j)$. If $i < i_*$, then $i_* \leftarrow i$. If $j > j_*$, then $j_* \leftarrow j$. Finally, pick $r \xleftarrow{\$} \mathcal{R}_\lambda$, run $\mathsf{ik}_{i \to j} \leftarrow \mathsf{Ext}(\mathsf{pko}, \mathsf{k}_*, i, j; r)$ and return $\mathsf{ik}_{i \to j}$.*

The IND-CCA*-advantage of $\mathcal{A}$ is defined by:*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{P,\mathcal{A}}(\lambda) = \left| \mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)] - 1/2 \right|.$$

The IND-CSCA experiment features an adversary who knows the public keys of a set of honest users and the public key of an honest opener. The adversary chooses two vectors of messages and a vector of public keys (of same size), then the experiment picks one of the two vectors and generates a vector of ciphertexts by encrypting each message using the corresponding public key and the encryption key of a designated honest user. For each encryption, if the public key belongs to an honest user, then the experiment encrypts the message of the picked vector, else it encrypts the message of the first vector (this disables the trivial attacks where the adversary receives messages of the picked vector encrypted with his own keys). The experiment also generates an interval key for the vector of ciphertexts using the opener public key and the designated user encryption key. The adversary receives the vector of ciphertexts and the interval key as challenge and tries to guess which vector has been encrypted.

The adversary has access to an encryption oracle that encrypts chosen messages using an index n incremented by the experiment after each encryption and the encryption key of the designated user, as well as a decryption oracle that decrypts any chosen ciphertext with the secret key of an honest user except the ciphertexts in the challenge. At the beginning of the experiment, the index n is initialized to 1, then n is incremented at each call to the encryption oracle, and at each encryption performed by the experiment in order to build the challenge. The adversary has access to an extraction oracle

that generates interval keys for chosen intervals using the designated user encryption key and the public key of the honest opener, except for the intervals that contain the index of one of the ciphertexts of the challenge. Note that without the secret key of the honest opener, the interval keys are supposed to be unusable by the adversary. Finally the adversary has access to an opening oracle that decrypts the ciphertexts on a chosen interval using a chosen vector of ciphertexts, a chosen interval key, and the secret key of the honest opener as the opening algorithm, except if the query exactly matches the challenge. The role of this oracle is more subtle than that of the decryption oracle : it allows the IND-CSCA security to prevent not only attacks where the adversary opens some modified ciphertexts from the challenge, but also attacks where the adversary adds or removes ciphertexts from the challenge, or reorders them before sending it to the opening oracle.

Definition 8 (μ-IND-CSCA experiment). *Let $P = (\mathcal{R}_\lambda, \mathcal{M}_\lambda, \mathsf{Gen}, \mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext}, \mathsf{Open})$ be an* APOPKE*, let λ be a security parameter and μ be an integer, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a pair of PPT algorithms. The* μ-Indistinguishability Against Chosen Set of Ciphertexts security *(*IND-CSCA*) experiment for P is defined by:*

$\mathbf{Exp}^{\mu\text{-IND-CSCA}}_{b,P,\mathcal{A}}(\lambda)$*:*

$n \leftarrow 1$*;*

$(\mathsf{pko}_*, \mathsf{sko}_*) \leftarrow \mathsf{Gen}(1^\lambda)$*;*

$\forall l \in [\![1, \mu]\!], (\mathsf{pk}_{(*,l)}, \mathsf{sk}_{(*,l)}) \leftarrow \mathsf{Gen}(1^\lambda)$*;*

$\mathsf{k}_* \leftarrow \mathsf{EGen}(1^\lambda)$*;*

$\mathcal{O} \leftarrow \left\{ \begin{array}{l} \mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n); \mathsf{Dec}(\cdot, \cdot); \\ \mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot); \mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot); \end{array} \right\}$;

$(j_*, (m_{(0,l)}, m_{(1,l)}, \bar{\mathsf{pk}}_l)_{n \le l \le j_*}, \mathsf{st}_*) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\lambda, \mathsf{pko}_*, (\mathsf{pk}_{(*,l)})_{1 \le l \le \mu})$*;*

$i_* \leftarrow n$*;*

$\forall\ l \in [\![i_*, j_*]\!]$*:* $r_{(*,l)} \xleftarrow{\$} \mathcal{R}_\lambda$*;*

If $\exists\ q, \mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$*, then* $c_{(*,l)} \leftarrow \mathsf{Enc}(\bar{\mathsf{pk}}_l, \mathsf{k}_*, m_{(b,l)}, l)$*;*

Else $c_{(*,l)} \leftarrow \mathsf{Enc}(\bar{\mathsf{pk}}_l, \mathsf{k}_*, m_{(0,l)}, l)$*;*

$n \leftarrow j_* + 1$*;*

$r_* \xleftarrow{\$} \mathcal{R}_\lambda$*;* $\mathsf{ik}_{i_* \to j_*} \leftarrow \mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, i_*, j_*; r_*)$*;*

$b_* \leftarrow \mathcal{A}_2^{\mathcal{O}}(\lambda, \mathsf{pko}_*, (\mathsf{pk}_{(*,l)})_{1 \le l \le \mu}, \mathsf{st}, \mathsf{ik}_{i_* \to j_*}, (c_{(*,l)})_{i_* \le l \le j_*})$*;*

If $b = b_*$, then return 1, else 0;

where the oracles are defined by:

$\underline{\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)}$**:** *On input (pk_n, m_n), pick $r_n \xleftarrow{\$} \mathcal{R}_\lambda$, run $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ and increment $n \leftarrow (n+1)$, then return c_n.*

$\underline{\mathsf{Dec}(\cdot, \cdot)}$**:** *On input (q, c), if $q \le 0$ or $q > \mu$ then return $\perp$. During the second phase if $\exists\ l$ such that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$ and $c_{(*,l)} = c$, then return $\perp$. Finally, run $m \leftarrow \mathsf{Dec}(\mathsf{sk}_{(*,q)}, c)$ and return m.*

$\underline{\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)}$**:** *On input (i, j), if $i \le 0$ or $i \ge j$ or $j \ge n$, then return $\perp$. During the second phase, if $[\![i, j]\!] \cap [\![i_*, j_*]\!] \neq \emptyset$, then return $\perp$. Finally, pick $r \xleftarrow{\$} \mathcal{R}_\lambda$, run $\mathsf{ik}_{i \to j} \leftarrow \mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, i, j; r)$ and return $\mathsf{ik}_{i \to j}$.*

$\underline{\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)}$**:** *On input $(i, j, \mathsf{ik}_{i \to j}, (c_l, \mathsf{pk}_l)_{i \le l \le j})$, if $i \le 0$ or $i \ge j$ or $j \ge n$, then return $\perp$. During the second phase, if $(\mathsf{ik}_{i \to j}, (c_l, \mathsf{pk}_l)_{i \le l \le j}) = (\mathsf{ik}_{i_* \to j_*}, (c_{(*,l)}, \bar{\mathsf{pk}}_l)_{i_* \le l \le j_*})$, then return $\perp$. Finally, run $(m_l)_{i \le l \le j} \leftarrow \mathsf{Open}(\mathsf{sko}_*, i, j, \mathsf{ik}_{i \to j}, (c_l, \mathsf{pk}_l)_{i \le l \le j})$ and return $(m_l)_{i \le l \le j}$.*

*The μ-*IND-CSCA*-advantage of $\mathcal{A}$ is defined by:*

$$\mathbf{Adv}^{\mu\text{-IND-CSCA}}_{P,\mathcal{A}}(\lambda) = \left| \mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mu\text{-IND-CSCA}}_{b,P,\mathcal{A}}(\lambda)] - 1/2 \right|.$$

Finally, we recall the *integrity* experiment of [5]. This property ensures that for the same ciphertext produced by an adversary, an honest receiver and an honest opener do not decrypt different plaintexts. This property should not be confused with the correctness, which ensures that a ciphertext will be decrypted correctly by the decryption algorithms only if it is generated by an honest user.

Definition 9 (Integrity experiment). *Let $P = (\mathcal{R}_\lambda, \mathcal{M}_\lambda, \mathsf{Gen}, \mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext}, \mathsf{Open})$ be an* APOPKE*, let λ be a security parameter, and let $\mathcal{A}$ be a PPT algorithm. We define the* integrity *(*Integrity*) experiment for P as follows:*

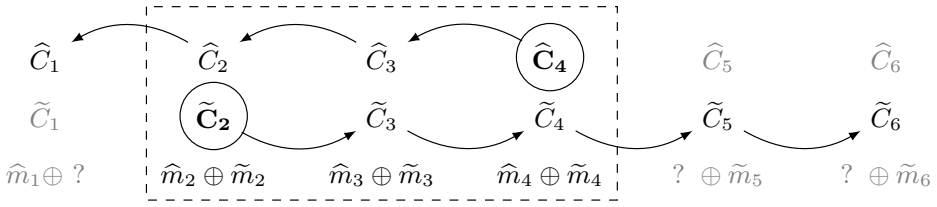

Fig. 1. Opening mechanism for the interval $[2, 4]$

$\mathbf{Exp}^{\mathsf{Integrity}}_{P,\mathcal{A}}(\lambda)$:
$(\mathsf{pko}_*, \mathsf{sko}_*) \leftarrow \mathsf{Gen}(1^\lambda)$;
$(i, j, (c_l, \mathsf{pk}_l)_{i \le l \le j}, x, \mathsf{sk}_x, \mathsf{ik}_{i \to j}) \leftarrow \mathcal{A}(\lambda, \mathsf{pko}_*)$;
$(m_l)_{i \le l \le j} \leftarrow \mathsf{Open}(\mathsf{sko}_*, i, j, \mathsf{ik}_{i \to j}, (c_l, \mathsf{pk}_l)_{i \le l \le j})$;
$m'_x \leftarrow \mathsf{Dec}(\mathsf{sk}_x, c_x)$;
If $m_x \neq \perp$ *and* $m'_x \neq \perp$ *and* $m_x \neq m'_x$ *and* $1 = \mathsf{KVer}(\mathsf{pk}_x, \mathsf{sk}_x)$,
then return 1, *else* 0;

The Integrity*-advantage is defined by* $\mathbf{Adv}^{\mathsf{Integrity}}_{P,\mathcal{A}}(\lambda) = \mathsf{Pr}[\mathbf{Exp}^{\mathsf{Integrity}}_{P,\mathcal{A}}(\lambda) = 1]$.

4 Our Scheme: CHAPO

Designing APOPKE schemes. CHAPO is based on the opening technique introduced in GAPO [5]: each message m_l is split in two parts $\widehat{m}_l$ and $\widetilde{m}_l$ such that $m_l = \widehat{m}_l \oplus \widetilde{m}_l$. The encryption algorithm uses four indexed keys denoted $\widehat{\mathsf{ek}}_l$, $\widehat{\mathsf{ek}}_{l-1}$, $\widetilde{\mathsf{ek}}_l$ and $\widetilde{\mathsf{ek}}_{l+1}$, and encrypts the message m_l in two blocs $\widehat{C}_l$ and $\widetilde{C}_l$ such that the part $\widehat{C}_l$ encrypts $\widehat{m}_l$ and $\widehat{\mathsf{ek}}_{l-1}$, and can be decrypted by the key $\widehat{\mathsf{ek}}_l$, and the part $\widetilde{C}_l$ encrypts $\widetilde{m}_l$ and $\widetilde{\mathsf{ek}}_{l+1}$, and can be decrypted by the key $\widetilde{\mathsf{ek}}_l$. The interval key for the interval $[i, j]$ is generated by encrypting the pair of keys $(\widetilde{\mathsf{ek}}_i, \widehat{\mathsf{ek}}_j)$ for the opener. Using $\widehat{\mathsf{ek}}_j$, the opener decrypts $\widehat{C}_j$ and obtains $\widehat{m}_j$ and $\widehat{\mathsf{ek}}_{j-1}$, then he uses $\widehat{\mathsf{ek}}_{j-1}$ to decrypt $\widehat{C}_{j-1}$ and repeats this operation recursively until he gets $(\widehat{m}_l)_{i \le l \le j}$. On the other hand, using $\widetilde{\mathsf{ek}}_i$, the opener decrypts $\widetilde{C}_i$ and obtains $\widetilde{m}_i$ and $\widetilde{\mathsf{ek}}_{i+1}$, then it uses $\widetilde{\mathsf{ek}}_{i+1}$ to decrypt $\widetilde{C}_{i+1}$ and repeats this operation recursively until he gets $(\widetilde{m}_l)_{i \le l \le j}$. Finally, he computes $(m_l)_{i \le l \le j} = (\widehat{m}_l \oplus \widetilde{m}_l)_{i \le l \le j}$.

We stress that the opener cannot decrypt the ciphertexts $\widetilde{C}_l$ for $l < i$ and $\widehat{C}_l$ for $l > j$, hence he is missing one share of each message m_l such that $l \notin [i, j]$, so he cannot rebuild these messages. The Figure 1 illustrates this mechanism for the interval $[2, 4]$. Note that this mechanism does not protect the messages between two openable intervals. However, it is possible to enlarge the interval over which the opener can decrypt the messages without altering the security of the scheme.

CHAPO overview. CHAPO uses a PRF, a MAC, a SKE and a PKE. The public/private keys $(\mathsf{pk}, \mathsf{sk})$ generation algorithm is defined as in the PKE scheme. The encryption key generator returns a secret pair $(\mathsf{rk}, \mathsf{mk})$ where rk is a PRF key and mk is a MAC key. For each index l, the keys $\widehat{\mathsf{ek}}_l$ and $\widetilde{\mathsf{ek}}_l$ are the outputs of the PRF on the (secret) key rk and the index l. Our scheme also requires a MAC key mk_l for each index l, that is also generated from the PRF on rk and l.

The encryption algorithm takes as input the encryption key $(\mathsf{rk}, \mathsf{mk})$ of the user, the message m_l and the public key pk of the message recipient. The algorithm first generates the ciphertext D_l by encrypting $[m_l \| \mathsf{mk}_l \| l]$ using the public key pk and a random coin denoted by v_l, then it splits m_l, mk_l, and v_l in two parts $(\widehat{m}_l, \widetilde{m}_l)$, $(\widehat{\mathsf{mk}}_l, \widetilde{\mathsf{mk}}_l)$, and $(\widehat{v}_l, \widetilde{v}_l)$, such that $m_l = \widehat{m}_l \oplus \widetilde{m}_l$, $\mathsf{mk}_l = \widehat{\mathsf{mk}}_l \oplus \widetilde{\mathsf{mk}}_l$, and $v_l = \widehat{v}_l \oplus \widetilde{v}_l$. The algorithm generates $\widetilde{C}_l$ by encrypting $[\widetilde{\mathsf{ek}}_{l+1} \| \widetilde{m}_l \| \widetilde{\mathsf{mk}}_l \| \widetilde{v}_l \| \mathsf{pk} \| l]$ with the key $\widetilde{\mathsf{ek}}_l$ using the SKE, and $\widehat{C}_l$ by encrypting $[\widehat{\mathsf{ek}}_{l-1} \| \widehat{m}_l \| \widehat{\mathsf{mk}}_l \| \widehat{v}_l \| \mathsf{pk} \| l]$ with the key $\widehat{\mathsf{ek}}_l$ using the SKE. Finally, the algorithm generates the MAC tag S_l using the key mk on $[\widetilde{C}_l \| \widehat{C}_l \| D_l \| \mathsf{pk} \| l]$ and the MAC tag T_l using mk_l on $[\widetilde{C}_l \| \widehat{C}_l \| D_l \| S_l \| \mathsf{pk} \| l]$. The ciphertext is the tuple $(\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$.

The recipient of $(\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$ decrypts D_l using sk in order to obtain the message m_l and the key mk_l, then it checks the MAC tag T_l using mk_l. The interval key extraction algorithm on the interval $[i, j]$ returns mk, $\widetilde{\mathsf{ek}}_i$, and $\widehat{\mathsf{ek}}_j$ encrypted with the public key of the opener. Using these keys, the opener recovers the messages $(m_l)_{i \le l \le j}$ by running the algorithm described above on the parts $\widetilde{C}_l$ and $\widehat{C}_l$ of each

ciphertext. The opener also verifies the tags S_l, and verifies the correctness of D_l ans T_l by regenerating them.

Definition 10 (CHAPO). *Let λ be a security parameter.* $\mathsf{CHAPO} = (\mathcal{R}_\lambda, \mathcal{M}_\lambda, \mathsf{Gen}, \mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext}, \mathsf{Open})$ *is an* APOPKE *scheme instantiated by a set $\mathcal{M}_\lambda$, an (unforgeable) message authentication code scheme* $\mathsf{MAC} = (\mathcal{K}^m_\lambda, \mathsf{Mac}, \mathsf{Ver})$*, an (IND-CCA secure) symmetric encryption scheme* $\mathsf{SKE} = (\mathcal{K}^s_\lambda, \mathcal{R}^s_\lambda, \mathsf{SEnc}, \mathsf{SDec})$*, a pseudorandom function* $\mathsf{PRF} : \mathcal{K}^r_\lambda \times \{0,1\}^{\lambda+1} \to \mathcal{K}^s_\lambda$*, and an (IND-CCA secure) public key encryption scheme* $\mathsf{PKE} = (\mathcal{R}^p_\lambda, \mathcal{M}^p_\lambda, \mathsf{PGen}, \mathsf{PEnc}, \mathsf{PDec})$*.* CHAPO *is defined by:*

$\underline{\mathsf{Gen}(1^\lambda)}$**:** *Run* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PGen}(1^\lambda)$ *and return* $(\mathsf{pk}, \mathsf{sk})$*.*

$\underline{\mathsf{EGen}(1^\lambda)}$**:** *Pick* $(\mathsf{rk}, \mathsf{mk}) \xleftarrow{\$} \mathcal{K}^r_\lambda \times \mathcal{K}^m_\lambda$*, set* $\mathsf{k} \leftarrow (\mathsf{rk}, \mathsf{mk})$ *and return* k*.*

$\underline{\mathsf{Enc}(\mathsf{pk}_l, \mathsf{k}, m_l, l; r_l)}$**:** *Parse* k *as* $(\mathsf{rk}, \mathsf{mk})$*. Using the random coin r_l, pick at random* $(\widehat{u}_l, \widetilde{u}_l) \xleftarrow{\$} (\mathcal{R}^s_\lambda)^2$*,* $v_l \xleftarrow{\$} \mathcal{R}^p_\lambda$*,* $\mathsf{mk}_l \xleftarrow{\$} \mathcal{K}^m_\lambda$*,* $\widehat{\mathsf{mk}}_l \xleftarrow{\$} \{0,1\}^{|\mathsf{mk}_l|}$*,* $\widehat{v}_l \xleftarrow{\$} \{0,1\}^{|v_l|}$*, and* $\widehat{m}_l \xleftarrow{\$} \{0,1\}^{|m_l|}$*.*
Set $\widetilde{\mathsf{mk}}_l \leftarrow \widehat{\mathsf{mk}}_l \oplus \mathsf{mk}_l$*,* $\widetilde{v}_l \leftarrow \widehat{v}_l \oplus v_l$*, and* $\widetilde{m}_l \leftarrow \widehat{m}_l \oplus m_l$*.*
$\forall\ h \in \{l-1, l, l+1\}$*, run* $\widehat{\mathsf{ek}}_h \leftarrow \mathsf{PRF}(\mathsf{k}, 0\|h)$ *and* $\widetilde{\mathsf{ek}}_h \leftarrow \mathsf{PRF}(\mathsf{k}, 1\|h)$*. Set:*
- $\widetilde{C}_l \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_l, [\widetilde{\mathsf{ek}}_{i+1}\|\widetilde{m}_l\|\widetilde{\mathsf{mk}}_l\|\widetilde{v}_l\|\mathsf{pk}_l\|l]; \widetilde{u}_l)$ *and* $\widehat{C}_l \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_l, [\widehat{\mathsf{ek}}_{i-1}\|\widehat{m}_l\|\widehat{\mathsf{mk}}_l\|\widehat{v}_l\|\mathsf{pk}_l\|l]; \widehat{u}_l)$
- $D_l \leftarrow \mathsf{PEnc}(\mathsf{pk}_l, [m_l\|\mathsf{mk}_l\|l]; v_l)$
- $S_l \leftarrow \mathsf{Mac}(\mathsf{mk}, [\widetilde{C}_l\|\widehat{C}_l\|D_l\|\mathsf{pk}_l\|l])$ *and* $T_l \leftarrow \mathsf{Mac}(\mathsf{mk}_l, [\widetilde{C}_l\|\widehat{C}_l\|D_l\|S_l\|\mathsf{pk}_l\|l])$

Set $c_l \leftarrow (\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, i)$ *and return* c_l*.*

$\underline{\mathsf{Dec}(\mathsf{sk}_l, c_l)}$**:** *Parse* c_l *as* $(\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$ *and run* $[m_l\|\mathsf{mk}_l\|l'] \leftarrow \mathsf{PDec}(\mathsf{sk}_l, D_l)$*. If* $l = l'$ *and* $\mathsf{Ver}(\mathsf{mk}_l, T_l, [\widetilde{C}_l\|\widehat{C}_l\|D_l\|S_l\|\mathsf{pk}_l\|l]) = 1$*, then return* m_l*, else return* $\perp$*.*

$\underline{\mathsf{Ext}(\mathsf{pko}, \mathsf{k}, i, j; r)}$**:** *Parse* k *as* $(\mathsf{rk}, \mathsf{mk})$*. Using the random coin r, pick at random* $u \xleftarrow{\$} \mathcal{R}^p_\lambda$ *Run* $\widehat{\mathsf{ek}}_j \leftarrow \mathsf{PRF}(\mathsf{rk}, 0\|j)$ *and* $\widetilde{\mathsf{ek}}_i \leftarrow \mathsf{PRF}(\mathsf{rk}, 1\|i)$*, then run* $X_{i\to j} \leftarrow \mathsf{PEnc}(\mathsf{pko}, [\widetilde{\mathsf{ek}}_i\|\widehat{\mathsf{ek}}_j\|\mathsf{mk}]; u)$ *and* $Y_{i\to j} \leftarrow \mathsf{Mac}(\mathsf{mk}, [i\|j\|X_{i\to j}])$*. Set* $\mathsf{ik}_{i\to j} \leftarrow (X_{i\to j}, Y_{i\to j}, i, j)$ *and return* $\mathsf{ik}_{i\to j}$*.*

$\underline{\mathsf{Open}(\mathsf{sko}, i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\le l\le j})}$**:** *Parse* $\mathsf{ik}_{i\to j}$ *as* $(X_{i\to j}, Y_{i\to j}, i', j')$*, run* $[\widetilde{\mathsf{ek}}_i\|\widehat{\mathsf{ek}}_j\|\mathsf{mk}] \leftarrow \mathsf{PDec}(\mathsf{sko}, X_{i\to j})$*.*
If $1 \neq \mathsf{Ver}(\mathsf{mk}, Y_{i\to j}, [i\|j\|X_{i\to j}])$ *or* $(i,j) \neq (i',j')$*, then abort and return* $\perp$*.*
$\forall\ l \in [\![i, j]\!]$*, from i to j, parse c_l as* $(\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$ *and run* $[\widetilde{\mathsf{ek}}_{l+1}\|\widetilde{m}_l\|\widetilde{\mathsf{mk}}_l\|\widetilde{v}_l\|\widetilde{\mathsf{pk}}_l\|\widetilde{l}] \leftarrow \mathsf{SDec}(\widetilde{\mathsf{ek}}_l, \widetilde{C}_l)$*.*
$\forall\ l \in [\![i, j]\!]$*, from j to i, run* $[\widehat{\mathsf{ek}}_{l-1}\|\widehat{m}_l\|\widehat{\mathsf{mk}}_l\|\widehat{v}_l\|\widehat{\mathsf{pk}}_l\|\widehat{l}] \leftarrow \mathsf{SDec}(\widehat{\mathsf{ek}}_l, \widehat{C}_l)$*. If:*
- $\widehat{\mathsf{pk}}_l \neq \mathsf{pk}_l$ *or* $\widehat{l} \neq l$ *or* $\widehat{\mathsf{pk}}_l \neq \mathsf{pk}_l$ *or* $\widehat{l} \neq l$*, or*
- $D_l \neq \mathsf{PEnc}(\mathsf{pk}_l, [\widetilde{m}_l \oplus \widehat{m}_l\|\widetilde{\mathsf{mk}}_l \oplus \widehat{\mathsf{mk}}_l\|l]; (\widetilde{v}_l \oplus \widehat{v}_l))$*, or*
- $T_l \neq \mathsf{Mac}((\widetilde{\mathsf{mk}}_l \oplus \widehat{\mathsf{mk}}_l), [\widetilde{C}_l\|\widehat{C}_l\|D_l\|S_l\|\mathsf{pk}_l\|l])$*, or*
- $1 \neq \mathsf{Ver}(\mathsf{mk}, S_l, [\widetilde{C}_l\|\widehat{C}_l\|D_l\|\mathsf{pk}_l\|l])$*,*

then return $\perp$*.*
Finally, return $(\widetilde{m}_l \oplus \widehat{m}_l)_{i\le l\le j}$*.*

Moreover, let le *be a function such that for any set S,* $\mathsf{le}(S) = \max_{x\in S}(|x|)$*, the set $\mathcal{R}_\lambda$ is defined by* $\mathcal{R}_\lambda = (\mathcal{R}^s_\lambda)^2 \times \mathcal{R}^p_\lambda \times \mathcal{K}^m_\lambda \times \{0,1\}^{\mathsf{le}(\mathcal{K}^m_\lambda)} \times \{0,1\}^{\mathsf{le}(\mathcal{R}^s_\lambda)} \times \{0,1\}^{\mathsf{le}(\mathcal{R}^s_\lambda)} \times \{0,1\}^{\mathsf{le}(\mathcal{M}_\lambda)}$*, and the set $\mathcal{M}^p_\lambda$ verifies* $\{0,1\}^{(\mathsf{le}(\mathcal{M}_\lambda)+\mathsf{le}(\mathcal{K}^m_\lambda)+\lambda)} \subseteq \mathcal{M}^p_\lambda$*.*

Security of CHAPO. The CCA security is achieved when the decryption oracles give no advantage to the adversary. The idea being that the decryption oracles will reject the ciphertexts that have not been produced by the experiment, or that have not been encrypted correctly by the adversary. In particular, the decryption oracles should reject any alteration of the challenge.

In the IND-CCA scenario, if the adversary alters the challenge c_{l_*}, then the tag T_{l_*} is no longer valid, and should be refreshed by the adversary. Hence, the adversary should recover the corresponding MAC key mk_{l_*}. This key is encrypted in $\widetilde{C}_{l_*}$ and $\widehat{C}_{l_*}$, however the adversary can decrypt only one of these two ciphertexts. Indeed, the challenge cannot be in the interval that the adversary can decrypt using his interval keys, according to the winning conditions of the IND-CCA experiment. In this case, one of the shares of mk_{l_*} remains encrypted by the SKE. mk_{l_*} is also encrypted in D_{l_*}, but since SKE and PKE are IND-CCA, the adversary cannot recover mk_{l_*} from $\widetilde{C}_{l_*}$ and $\widehat{C}_{l_*}$, or from D_{l_*}. Moreover, since the adversary does not know the key rk, he cannot generate mk_{l_*} from the PRF. Finally, the adversary cannot alter the challenge c_{l_*}.

In the IND-CSCA scenario, if the adversary alters a ciphertext c in the challenge, then the tag T is no longer valid for the key mk, and should be refreshed by the adversary. Otherwise, the opening oracle

will fail on any vector of ciphertexts that contains the alteration of c. However, the adversary has not the key mk. To recover it, the adversary would have to decrypt one of the interval keys without the secret key sko of the honest opener, which is not possible since the PKE scheme is IND-CCA secure. Similarly, the opening oracle rejects any query that contains new ciphertexts created by the adversary because he cannot generate the corresponding MAC tags. On the other hand, the adversary could try to use the opening oracle on a vector that contains the ciphertexts of the challenge in a new order. However, since the MAC tags authenticate the indexes of the ciphertexts, the opening oracle will always fail on such a query. Note that since there is only one opener in our model, using a single MAC key mk for all the ciphertexts produced by the same user does not lead to security problem: the key mk can be seen as an authentication key shared between the user who encrypts the messages and the opener.

Finally, when the opener decrypts the ciphertext $c_l = (\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$ encrypted with a key pk_l in some interval, he recovers the message m_l and learns the values mk_l and v_l, then he checks that $D_l = \mathsf{PEnc}(\mathsf{pk}_l, [m_l \| \mathsf{mk}_l \| l]; v_l)$, which ensures that $[m_l \| \mathsf{mk}_l \| l] = \mathsf{PDec}(\mathsf{sk}_l, D_l)$. This implies that the decryption algorithm returns the same message m_l as the opening algorithm, which shows that CHAPO has the integrity property.

Comparaison with GAPO. CHAPO is based on the same approach as GAPO. However, the design of GAPO does not achieve the functionality required by CHAPO, and the two schemes differ in several points. First, the extraction algorithm of GAPO takes as input the first and last ciphertexts of the interval instead of using the corresponding indices, and the encryption algorithm uses a *secret state* that is updated with each encryption. This implies that the user must have access to all the ciphertexts they sent, and that their devices must synchronize to share the secret state in a secure way, which limits the practicality of GAPO.

GAPO uses a PKE family called *Random Coin Decryptable* (RCD) PKE, which means that the ciphertext can be decrypted with its random coin in addition to the standard decryption via the secret key. A GAPO ciphertext consists of two RCD ciphertexts. The first one contains half of the message, and the (symmetric) encryption of the seed of the random coin and the (symmetric) key used for the next encryption, and the second one contains the other half of the message, and the (symmetric) encryption of the seed of the random coin and the (symmetric) key used by the previous encryption. This method makes it very easy to achieve integrity because the opener and the receiver decrypt the same ciphertext. On the other hand, it is slightly less generic since it uses a specific family of encryption, and it requires two public key encryptions, doubling the encryption time of GAPO compared to CHAPO. Moreover, the random coins are obtained by hashing the seeds, and the hash function must be modeled by a random oracle so that the coins are *truly random*. This makes GAPO's security highly relying on the random oracle. Finally, GAPO is not CCA secure by design. To achieve this security, CHAPO uses messages authentication codes, as explained in the previous paragraph.

Security proofs. We prove each theorem by using a sequence of games between a *challenger* and an *adversary* where the adversary plays an experiment run by the challenger. Due to page limitation, we only give the sequences of games, but we omit the reductions between the games. The full proofs are given in Appendix A.

Theorem 1. *If* PRF *is pseudorandom,* MAC *is* EUF-CMA*,* SKE *is* IND-CCA*, and* PKE *is correct and* IND-CCA*, then* CHAPO *is* IND-CSCA*. Moreover, the following holds, where q_n (resp. q_x) denotes the number of queries to the oracle* Enc *(resp.* Ext*) during the experiment, and q_* is an upper bound on $j_* - i_*$:*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{IND\text{-}CSCA}}_{\mathsf{CHAPO}}(\lambda) &\leq 2 \cdot \left(q_x + q_* \cdot \mu \cdot (1 + (q_n + q_*)^2)\right) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda) \\ &+ (2 + (q_n + q_*)^2 \cdot q_*) \cdot \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda) + 4 \cdot (q_n + q_*) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda) \\ &+ \mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF}}(\lambda).\end{aligned}$$

Proof (sketch). Let $\mathcal{A}$ be a PPT algorithm, we use the following sequence of games:
Game G_0: This game is the original IND-CSCA experiment.
Game G_1: This game is similar to G_0, but the challenger replaces each output of the PRF by a random value picked at random in $\mathcal{K}^s_\lambda$. We prove by reduction that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]| \leq \mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF}}(\lambda).$$

Game G_2: Let q_x be the number of queries sent to the oracle $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$. This game is similar to G_1, but the challenger replaces the elements encrypted in the interval keys by random values:

- At the beginning of the experiment, the challenger initializes an empty list $\mathcal{L}_X[\,]$.
- Each time that the oracle $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$ receives a query (i, j), the challenger parses k_* as $(\mathsf{rk}_*, \mathsf{mk}_*)$. If $i \leq 0$, or $i \geq j$, or $j \geq n$, or $[\![i, j]\!] \cap [\![i_*, j_*]\!] \neq \emptyset$, then it returns $\perp$ according to the definition of the oracle Ext. It then picks $r \xleftarrow{\$} \mathcal{R}^p_\lambda$ and $\mathsf{rnd} \xleftarrow{\$} \{0,1\}^{|\widetilde{\mathsf{ek}}_i \| \widehat{\mathsf{ek}}_j \| \mathsf{mk}_*|}$, runs $X_{i\to j} \leftarrow \mathsf{PEnc}(\mathsf{pko}_*, \mathsf{rnd}; r)$, sets $\mathcal{L}_{\mathsf{Ext}}[X_{i\to j}] \leftarrow [i\|j]$, runs $Y_{i\to j} \leftarrow \mathsf{Mac}(\mathsf{mk}_*, [i\|j\|X_{i\to j}])$, sets $\mathsf{ik}_{i\to j} \leftarrow (X_{i\to j}, Y_{i\to j}, i, j)$ and returns $\mathsf{ik}_{i\to j}$.
- Each time that the challenger runs $\mathsf{Open}(\mathsf{sko}_*, i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\leq l\leq j})$ such that, parsing $\mathsf{ik}_{i\to j}$ as $(X_{i\to j}, Y_{i\to j}, i, j)$, $\mathcal{L}_X[X_{i\to j}] \neq \perp$, it replaces the instruction $[\widetilde{\mathsf{ek}}'_i \| \widehat{\mathsf{ek}}'_j \| \mathsf{mk}'] \leftarrow \mathsf{PDec}(\mathsf{sko}_*, X_{i\to j})$ by the instructions $[i'\|j'] \leftarrow \mathcal{L}_X[X_{i\to j}]$; $\widetilde{\mathsf{ek}}'_i \leftarrow \widetilde{\mathsf{ek}}_{i'}$; $\widehat{\mathsf{ek}}'_j \leftarrow \widehat{\mathsf{ek}}_{j'}$; $\mathsf{mk}' \leftarrow \mathsf{mk}_*$.

We prove by reduction that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_1\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_2\right]| \leq 2 \cdot q_x \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

We stress that from this game, **the key mk_* is never used by the challenger except to generate/verify the message authentication codes**.

Game G_3: In this game, the challenger aborts and returns a random bit if the adversary sends a valid query with a fresh interval key $\mathsf{ik}_{i\to j}$ to the oracle Open, *i.e.* an interval key that has not been generated by the experiment and that does not abort the oracle. more precisely, this game is similar to G_2, but:

- At the beginning, the challenger initializes an empty set $\mathcal{S}_{\mathsf{ik}}$.
- Each time that the challenger generates an interval key $\mathsf{ik}_{i\to j}$, the challenger sets $\mathcal{S}_{\mathsf{ik}} \leftarrow \mathcal{S}_{\mathsf{ik}} \cup \{\mathsf{ik}_{i\to j}\}$.
- Each time that the oracle $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ receives a query $(i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\leq l\leq j})$ such that the oracle does not returns $\perp$, if parsing $\mathsf{ik}_{i\to j}$ as $(X_{i\to i}, Y_{i\to i}, i, j)$ it holds that $\mathsf{ik}_{i\to j} \notin \mathcal{S}_{\mathsf{ik}}$ and $1 = \mathsf{Ver}(\mathsf{mk}_*, Y_{i\to j}, [i\|j\|X_{i\to i}])$, then the challenger sets $\mathsf{Abort}_3 \leftarrow 1$, aborts the experiment and returns a random bit.

We claim that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_2\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right]| \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

We have that $|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_2\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right]| \leq \mathsf{Pr}[\mathsf{Abort}_3 = 1]$. We prove this claim by showing that $\mathsf{Pr}[\mathsf{Abort}_3 = 1] \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda)$.

Game G_4: In this game, the challenger aborts and returns a random bit if the adversary sends a valid query with a fresh ciphertext c_l to the oracle Open, *i.e.* a ciphertext that has not been generated by the experiment and that does not abort the oracle. more precisely, this game is similar to G_3, but:

- At the beginning of the experiment, the challenger initializes an empty set $\mathcal{S}_{\mathsf{Enc}}$.
- Each time that the challenger runs the encryption algorithm $c \leftarrow \mathsf{Enc}(\mathsf{pk}_{(*,q)}, \mathsf{k}_*, m, l; r)$ for any input m and l, and any key index q, the challenger sets $\mathcal{S}_{\mathsf{Enc}} \leftarrow \mathcal{S}_{\mathsf{Enc}} \cup \{(\mathsf{pk}_{(*,q)}, c)\}$.
- Each time that the oracle $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ receives a query $(i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\leq l\leq j})$ such that $\exists\ l$ such that parsing c_l as $(\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$, it holds that $(\mathsf{pk}_l, c_l) \notin \mathcal{S}_{\mathsf{Enc}}$ and $1 = \mathsf{Ver}(\mathsf{mk}_*, S_l, [\widetilde{C}_l \| \widehat{C}_l \| D_l \| \mathsf{pk}_l \| l])$, and such that the oracle do not return $\perp$, the challenger sets $\mathsf{Abort}_4 \leftarrow 1$, aborts the experiment and returns a random bit.

We claim that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_4\right]| \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

We have that $|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_4\right]| \leq \mathsf{Pr}[\mathsf{Abort}_4 = 1]$. We prove this claim by showing that $\mathsf{Pr}[\mathsf{Abort}_4 = 1] \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda)$.

Game G_5: In what follows, c_l denotes the encryption of the l^{th} message m_l by the experiment, and $\overline{\mathsf{pk}}_l$ denotes the corresponding public key. Moreover, for any pair of indexes (i, j), $\mathsf{ik}^h_{i\to j}$ denotes the h^{th} interval key returned by the oracle $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$ on the input (i, j). This game is the same as G_5 but the oracle $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ is re-defined by:

- On input $(i, j, \mathsf{ik}'_{i\to j}, (c'_l, \mathsf{pk}'_l)_{i\leq l\leq j})$, if $i \leq 0$ or $i \geq j$ or $j \geq n$, then return $\perp$. During the second phase, if $(\mathsf{ik}'_{i\to j}, (c'_l, \mathsf{pk}'_l)_{i\leq l\leq j}) = (\mathsf{ik}_{i_*\to j_*}, (c_{(*,l)}, \overline{\mathsf{pk}}_l)_{i_*\leq l\leq j_*})$, then return $\perp$.
- If $(c'_l, \mathsf{pk}'_l)_{i\leq l\leq j} = (c_l, \mathsf{pk}_l)_{i\leq l\leq j}$ and $\exists\ h$ such that $\mathsf{ik}'_{i\to j} = \mathsf{ik}^h_{i\to j}$, then return $(m_l)_{i\leq l\leq j}$, else return $\perp$.

From the properties of G_4 given above, we deduce that:

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_4\right] = \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_5\right].$$

We stress that from this game, **the challenger no longer use the algorithm** SDec **during the experiment**.

Game G_6**:** In this game, the challenger replaces the message encrypted in each ciphertext $\widetilde{C}_l$ by a random value. More precisely, this game is similar to G_5, but each time that the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ is run by the Enc oracle or by the experiment during the generation of the challenge, the challenger replaces the instruction $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]; \widetilde{u}_n)$ by the sequence of instructions $\mathsf{str}_n \leftarrow [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, \mathsf{rnd}_n; \widetilde{u}_n)$;
Let q_* be an upper bound on the number of ciphertexts sending by the adversary $\mathcal{A}_1$, *i.e.* $j_* - i_* \leq q_*$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_5] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_6]| \leq 2 \cdot (q_n + q_*) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

We prove this claim by using an hybrid argument. We define an hybrid game $G_{6,h}$ as follows:

Game $G_{6,h}$**:** In this game, the challenger replaces the messages encrypted in the h first ciphertexts $\widetilde{C}_l$ by random values. More precisely, If $h = 0$, then $G_{6,h} = G_5$, else if $1 \leq h \leq (q_n + q_*)$, then the game $G_{6,h}$ is the same as $G_{6,h-1}$, but when the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ is run for $n = h$ by the Enc oracle or by the experiment during the generation of the challenge, the challenger replaces the instructions $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]; \widetilde{u}_n)$ by $\mathsf{str}_n \leftarrow [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, \mathsf{rnd}_n; \widetilde{u}_n)$. We prove by reduction that, for all $h \in [\![1, q_n + q_*]\!]$:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{6,h-1}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{6,h}]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

Game G_7**:** In this game, the challenger replaces the message encrypted in each ciphertext $\widehat{C}_l$ by a random value. More precisely, this game is similar to G_6, but when the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ is run for $n = h$ by the Enc oracle or by the experiment during the generation of the challenge, the challenger replaces the instructions $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]; \widehat{u}_n)$ by the sequence $\mathsf{str}_n \leftarrow [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, \mathsf{rnd}_n; \widehat{u}_n)$. Let q_* be an uper bound on the number of ciphertexts sending by the adversary $\mathcal{A}_1$, *i.e.* $j_* - i_* \leq q_*$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_6] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]| \leq 2 \cdot (q_n + q_*) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

This claim can be proved using the following hybrid argument: We define an hybrid game $G_{7,h}$ as follows:

Game $G_{7,h}$**:** In this game, the challenger replaces the messages encrypted in the $(q_n + q_* + 1 - h)$ last ciphertexts $\widetilde{C}_l$ by random values. More precisely, if $h = q_n + q_* + 1$, then $G_{7,h} = G_6$, else if $1 \leq h \leq (q_n + q_*)$, then the game $G_{7,h}$ is the same as $G_{7,h+1}$, but each time the challenger runs the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ during the Enc oracle or during the challenge generation such that $n = h$, it replaces the instruction $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]; \widehat{u}_n)$ by the sequence $\mathsf{str}_n \leftarrow [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, \mathsf{rnd}_n; \widehat{u}_n)$. We prove by reduction that, for all $h \in [\![1, q_n + q_*]\!]$:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{7,h+1}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{7,h}]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

Game G_8**:** In what follows, we parse the l^{th} ciphertext $c_{(*,l)}$ of the challenge by $(\widetilde{C}_{(*,l)}, \widehat{C}_{(*,l)}, D_{(*,l)}, S_{(*,l)}, T_{(*,l)}, l)$. In this game, the challenger replaces the keys mk_l for all l in $[\![i_*, j_*]\!]$ encrypted by a public key generated by the challenger (*i.e.* in $\{\mathsf{pk}_{(*,q)}\}_{1 \leq q \leq \mu}$) in each ciphertext $D_{(*,l)}$ by a random value. More precisely, this game is similar to G_7, except that when the challenger build the challenge $(\mathsf{ik}_{i_* \to j_*}, (c_{(*,l)})_{i_* \leq l \leq j_*})$, $\forall\, l \in [\![i_*, j_*]\!]$ such that $\exists\, q, \mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$:

- the challenger picks $\mathsf{mk}'_l \xleftarrow{\$} \mathcal{K}^m_\lambda$,
- when the challenger runs $c_{(*,l)} \leftarrow \mathsf{Enc}(\bar{\mathsf{pk}}_l, \mathsf{k}_*, m_{(b,l)}, l)$, it replaces the instruction $D_{(*,l)} \leftarrow \mathsf{PEnc}(\mathsf{pk}_{(*,q)}, [m_{(b,l)} \| \mathsf{mk}_l \| l]; v_l)$ by $D_{(*,l)} \leftarrow \mathsf{PEnc}(\mathsf{pk}_{(*,q)}, [m_{(b,l)} \| \mathsf{mk}'_l \| l]; v_l)$.
- For each query (p, c) (where we parse c as $(\widetilde{C}, \widehat{C}, D, S, T, l')$) sending to the oracle $\mathsf{Dec}(\cdot, \cdot)$ such that $p = q$, $c \neq c_{(*,l)}$ and $D = D_{(*,l)}$, the challenger runs $\mathsf{Dec}(\mathsf{sk}_{(*,q)}, c)$ as in the real experiment except that it replaces the instruction $[m \| \mathsf{mk} \| l''] \leftarrow \mathsf{PDec}(\mathsf{sk}_{(*,q)}, D)$ by $[m \| \mathsf{mk} \| l''] \leftarrow [m_{(b,l)} \| \mathsf{mk}_l \| l]$.

We prove by reduction that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_7\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_8\right]| \leq 2 \cdot q_* \cdot \mu \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

We stress that from this game, for each index l in $[\![i_*, j_*]\!]$ such that $\bar{\mathsf{pk}}_l \in \{\mathsf{pk}_{(*,q)}\}_{1 \leq q \leq \mu}$, **the key mk_l is only used to produced the MAC tag $T_{(*,l)}$. Especially, mk_l is never encrypted in a ciphertext known by the adversary**.

Game G_9: In this game, the challenger tries to guess the indexes i_* and j_*, and aborts in case of failure. More precisely, this game is similar to G_8, but the challenger picks $(i'_*, j'_*) \leftarrow [\![1, q_n + q_*]\!]^2$ at the beginning of the experiment. At the end of the experiment, if $i'_* \neq j_*$ or $i'_* \neq j_*$, then the challenger returns a random bit. The adversary increases its winning advantage by a factor equalling the probability of guessing correctly i_* and j_*:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_8\right] - 1/2| = (q_n + q_*)^2 \cdot |\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_9\right] - 1/2|$$

We stress that n is incremented $(q_n + q_*)$ times during the experiment: q_n times by the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$ and q_* times after the generation of $(c_{(*,l)})_{i_* \leq l \leq j_*}$.

Game G_{10}: We parse the challenge $(c_{(*,l)})_{i_* \leq l \leq j_*}$ as $(\widetilde{C}_{(*,l)}, \widehat{C}_{(*,l)}, D_{(*,l)}, S_{(*,l)}, T_{(*,l)}, l)_{i_* \leq l \leq j_*}$. In this game, the challenger aborts if the adversary tries to reuse the element $D_{(*,l)}$ in a ciphertext $c \neq c_{(*,l)}$ sending to the decryption oracle for all l in $[\![i_*, j_*]\!]$ such that $\bar{\mathsf{pk}}_l \in \{\mathsf{pk}_{(*,q)}\}_{1 \leq q \leq \mu}$. More concretely, this game is similar to G_9, but if the adversary sends a query (p, c) (where we parse c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$) to the oracle $\mathsf{Dec}(\cdot, \cdot)$ such that, $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$ and $D = D_{(*,l)}$ and $c \neq c_{(*,l)}$ and $\mathsf{Ver}(\mathsf{mk}_l, T, [\widetilde{C} \| \widehat{C} \| D \| S \| \mathsf{pk}_{(*,q)} \| l]) = 1$, then the challenger set $\mathsf{Abort}_{10} \leftarrow 1$, aborts the game G_{10} and returns a random bit. We claim that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_9\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{10}\right]| \leq q_* \cdot \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

We prove this claim by using an hybrid argument. We define the game $G_{10,h}$ as follows:

Game $G_{10,h}$: We define $G_{10,0}$ as G_9, and for all $1 \leq h \leq q_*$, we define $G_{10,h}$ as $G_{10,h-1}$ but the challenger aborts if (i) $\bar{\mathsf{pk}}_{(*,i'_*+h)} \in \{\mathsf{pk}_{(*,q)}\}_{1 \leq q \leq \mu}$, and (ii) the adversary tries to reuse the element $D_{(*,i'_*+h)}$ in a ciphertext $c \neq c_{(*,i'_*+h)}$ sending to the decryption oracle. More concretely, we define $G_{10,h}$ as the same game as $G_{10,h-1}$ except that parsing the challenge $(c_{(*,l)})_{i_* \leq l \leq j_*}$ as $(\widetilde{C}_{(*,l)}, \widehat{C}_{(*,l)}, D_{(*,l)}, S_{(*,l)}, T_{(*,l)})_{i_* \leq l \leq j_*}$, if the adversary sends a query (p, c) to the oracle $\mathsf{Dec}(\cdot, \cdot)$ such that, parsing c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$, it holds that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_{i'_*+h}$ and $D = D_{(*,i'_*+h)}$ and $c \neq c_{(*,i'_*+h)}$ and $\mathsf{Ver}(\mathsf{mk}_{i'_*+h}, T, [\widetilde{C} \| \widehat{C} \| D \| S \| \mathsf{pk}_{(*,q)} \| l]) = 1$, then the challenger set $\mathsf{Abort}_{10,h} \leftarrow 1$, aborts the game $G_{10,h}$ and returns a random bit. We note that $G_{10,q_*} = G_{10}$. We prove by reduction that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{10,h-1}\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{10,h}\right]| \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

Game G_{11}: This game is similar to G_{10}, but the challenger substitutes each message that depends on b by a random message of same length. More formally, for each $l \in [\![i'_*, j'_*]\!]$ such that $\bar{\mathsf{pk}}_l \in \{\mathsf{pk}_{(*,q)}\}_{1 \leq q \leq \mu}$ it replaces the message $m_{(b,l)}$ encrypted in $c_{(*,l)}$ by the public key $\bar{\mathsf{pk}}_l$ by a message $m_{(*,l)} \xleftarrow{\$} \{0,1\}^{|m_{(b,l)}|}$ chosen at random. We prove by reduction that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{10}\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{11}\right]| = 2 \cdot q_* \cdot \mu \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

By composing the winning probabilities of $\mathcal{A}$ in all games, we obtain the upper bound on $\mathbf{Adv}^{\mathsf{IND\text{-}CSCA}}_{\mathsf{CHAPO}}(\lambda)$ given in the theorem, which concludes the proof. □

Theorem 2. *If* PRF *is pseudorandom,* MAC *is* EUF-CMA*,* SKE *and* PKE *are correct and* IND-CCA*, then* CHAPO *is* IND-CCA*. Moreover, the following holds, where q_n denotes the number of queries to the oracle* Enc *during the experiment:*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{CHAPO}}(\lambda) \leq & 2 \cdot \mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF}}(\lambda) + 4 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda) \\ & + 8 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda) + 2 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).\end{aligned}$$

Proof (sketch). We recall that in order to win the IND-CCA experiment with a non negligible advantage, the following must hold: $l_* \notin [\![i_*, j_*]\!]$. We separate our proof in two distinct cases: $l_* < i_*$ and $l_* > j_*$.

More concretely, we define two variants of the IND-CCA experiment: the $\mathsf{IND\text{-}CCA}_0$ (resp. $\mathsf{IND\text{-}CCA}_1$) experiment denotes the same experiment as IND-CCA except that if $l_* < i_*$ (resp. $l_* > j_*$), then the challenger returns a random bit. We have that:

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{CHAPO},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{IND\text{-}CCA}_0}_{\mathsf{CHAPO},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{IND\text{-}CCA}_1}_{\mathsf{CHAPO},\mathcal{A}}(\lambda).$$

Case $\mathsf{IND\text{-}CCA}_0$ (*i.e.* $l_* < i_*$): We use the following sequence of games. Let $\mathcal{A}$ be a PPT algorithm:
Game G_0: This game is the original $\mathsf{IND\text{-}CCA}_0$ experiment.
Game G_1: This game is similar to G_0, but the challenger replaces each output of the PRF by a random value picked in $\mathcal{K}^s_\lambda$. We prove by reduction that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_0\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_1\right]| \leq \mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF}}(\lambda).$$

Game G_2: In this game, the challenger tries to guess the index i_* and aborts in case of failure. Let q_n be the number of calls to the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$. This game is similar to G_1, but the challenger picks $i'_* \leftarrow [\![1, q_n + 1]\!]$ at the beginning of the experiment. At the end of the experiment, if $i'_* \neq i_*$, then the challenger returns a random bit. The adversary increases its winning advantage by a factor equalling the probability of guessing correctly i_*:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_1\right] - 1/2| = (q_n + 1) \cdot |\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_2\right] - 1/2|$$

Note that n is incremented $(q_n + 1)$ times during the experiment: q_n times by the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$ and one time after the generation of c_{l_*}.
Game G_3: In this game, for all $n \leq i_*$, the challenger replaces the part $\widetilde{C}_n$ of the ciphertext c_n by the encryption of a random message, which includes the challenge c_{l_*} if $l_* \leq i_*$ (otherwise, the challenger returns a random bit by definition of the $\mathsf{IND\text{-}CCA}_0$ experiment). This game is similar to G_2, but while $n < i'_*$, each time the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$ is called on a query (pk_n, m_n) and runs $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$, it replaces the instruction $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]; \widetilde{u}_n)$ by the sequence of instructions $\mathsf{str}_n \leftarrow [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, \mathsf{rnd}_n; \widetilde{u}_n)$. Moreover, when it computes the challenge $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,0)}, l_*; r_*)$, the challenger replaces the instruction $\widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, [\widetilde{\mathsf{ek}}_{l_*+1} \| \widetilde{m}_{(*,b)} \| \widetilde{\mathsf{mk}}_{l_*} \| \widetilde{v}_{l_*} \| \mathsf{pk}_* \| l_*]; \widetilde{u}_{l_*})$; by the sequence $\mathsf{str}_{l_*} \leftarrow [\widetilde{\mathsf{ek}}_{l_*+1} \| \widetilde{m}_{(*,b)} \| \widetilde{\mathsf{mk}}_{l_*} \| \widetilde{v}_{l_*} \| \mathsf{pk}_* \| l_*]$; $\mathsf{rnd}_{l_*} \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_{l_*}|}$; $\widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, \mathsf{rnd}_{l_*}; \widetilde{u}_{l_*})$. We claim that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_2\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right]| \leq 2 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

We prove this claim by using an hybrid argument. We define the hybrid game $G_{3,i}$ as follows:
Game $G_{3,i}$: If $i = 0$, then $G_{3,0} = G_2$, else for all $n \leq i$, if $i \leq i'_*$, then the challenger replaces the part $\widetilde{C}_n$ of the ciphertext c_n by the encryption of a random message. More concretely, if $1 \leq i \leq (q_n + 1)$, then the game $G_{3,i}$ is the same as $G_{3,i-1}$, but if $i < i'_*$ and $l_* \neq i$, then when the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, i)$ is called on a query (pk_i, m_i) and runs $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{k}_*, m_i, i; r_i)$, it replaces the instruction $\widetilde{C}_i \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_i, [\widetilde{\mathsf{ek}}_{i+1} \| \widetilde{m}_i \| \widetilde{\mathsf{mk}}_i \| \widetilde{v}_i \| \mathsf{pk}_i \| i]; \widetilde{u}_i)$ by the sequence of instructions $\mathsf{str}_i [\widetilde{\mathsf{ek}}_{i+1} \| \widetilde{m}_i \| \widetilde{\mathsf{mk}}_i \| \widetilde{v}_i \| \mathsf{pk}_i \| i]$; $\mathsf{rnd}_i \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_i|}$; $\widetilde{C}_i \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_i, \mathsf{rnd}_i; \widetilde{u}_i)$. Moreover, if $l_* = i$, then when it computes the challenge $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,0)}, l_*; r_*)$, the challenger replaces the instruction $\widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, [\widetilde{\mathsf{ek}}_{l_*+1} \| \widetilde{m}_{(*,b)} \| \widetilde{\mathsf{mk}}_{l_*} \| \widetilde{v}_{l_*} \| \mathsf{pk}_* \| l_*]; \widetilde{u}_{l_*})$ by $\mathsf{str}_{l_*} \leftarrow [\widetilde{\mathsf{ek}}_{l_*+1} \| \widetilde{m}_{(*,b)} \| \widetilde{\mathsf{mk}}_{l_*} \| \widetilde{v}_{l_*} \| \mathsf{pk}_* \| l_*]$; $\mathsf{rnd}_{l_*} \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_{l_*}|}$; $\widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, \mathsf{rnd}_{l_*}; \widetilde{u}_{l_*})$. We prove by reduction that, for all $i \in [\![1, q_n + 1]\!]$:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{3,i-1}\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{3,i}\right]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

Game G_4: This is the same game as G_3, but $\widehat{m}_{(*,b)}$, $\widehat{\mathsf{mk}}_{l_*}$ and $\widehat{v}_{l_*}$ are substituted by random bit-strings of same length. At this step, since $\widetilde{m}_{(*,b)}$ is replaced by a random bit string in $\widetilde{C}_{l_*}$ and $\widehat{m}_{(*,b)} = m_{(*,b)} \oplus \widetilde{m}_{(*,b)}$, then $\widehat{m}_{(*,b)}$ is indistinguishable from a random bit string, so it no longer depends on b, and can be substituted by a random bit-string without any influence on the adversary advantage. Using a similar argument, we have that $\widehat{\mathsf{mk}}_{l_*}$ and $\widehat{v}_{l_*}$ can also be substituted by random bit-strings. We deduce that:

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right] = \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_4\right].$$

Game G_5: In this game, the challenger replaces the MAC key mk_{l_*} by a random value in the part D_{l_*} (encrypted by PKE) of the ciphertext challenge c_{l_*}. More concretely, This game is similar to G_5, except that:

– when it computes the challenge by running $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,b)}, l_*; r_*)$, it replaces the instruction $D_{l_*} \leftarrow \mathsf{PEnc}(\mathsf{pk}_*, [m_{(*,b)} \| \mathsf{mk}_{l_*} \| l_*]; v_{l_*})$ by the sequence of instructions $\mathsf{mk}'_{l_*} \xleftarrow{\$} \mathcal{K}^m_\lambda; D_{l_*} \leftarrow \mathsf{PEnc}(\mathsf{pk}_*, [m_{(*,b)} \| \mathsf{mk}'_{l_*} \| l_*]; v_{l_*})$,
– For each query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ sending to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ such that $c \neq c_{l_*}$ and $D = D_{l_*}$, the challenger runs $\mathsf{Dec}(\mathsf{sk}_*, c)$ as in the real experiment except that it replaces the instruction $[m \| \mathsf{mk} \| l] \leftarrow \mathsf{PDec}(\mathsf{sk}_*, D)$ by $[m \| \mathsf{mk} \| l] \leftarrow [m_{(*,b)} \| \mathsf{mk}_{l_*} \| l_*]$.

We prove by reduction that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_4] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_5]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

Game G_6: In what follows, we parse the challenge c_{l_*} as $(\widetilde{C}_{l_*}, \widehat{C}_{l_*}, D_{l_*}, S_{l_*}, T_{l_*})$. In this game, the challenger aborts if the adversary tries to reuse D_{l_*} in a ciphertext $c \neq c_{l_*}$ sending to the decryption oracle. More concretely, this game is similar to G_5, but if the adversary sends a query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ such that $D = D_{l_*}$, $c \neq c_{l_*}$ and $\mathsf{Ver}(\mathsf{mk}_{l_*}, T, [\widetilde{C} \| \widehat{C} \| D \| S \| \mathsf{pk}_* \| l]) = 1$, then the challenger set $\mathsf{Abort}_6 \leftarrow 1$, aborts the game G_6 and returns a random bit. We prove by reduction that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_5] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_6]| \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

Game G_7: This game is similar to G_6, but the challenger substitutes the message $m_{(*,b)}$ by a random message $m_* \xleftarrow{\$} \{0,1\}^{|m_{(*,b)}|}$. We prove by reduction that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_6] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]| = 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

At this step, the parts $\widehat{C}_{l_*}$, $\widetilde{C}_{l_*}$, and D_{l_*} of the challenge c_{l_*} encrypts random values instead of the messages $\widehat{m}_{(*,b)}$, $\widetilde{m}_{(*,b)}$, and $m_{(*,b)}$, which implies that the game G_7 do not depend on the challenge bit b. We deduce that $\mathsf{Pr}[\mathcal{A} \text{ wins } G_7] = 1/2$. Case $\mathsf{IND\text{-}CCA}_1$ ($l_* > j_*$) is similar to case $\mathsf{IND\text{-}CCA}_0$. By composing the winning probabilities of $\mathcal{A}$ in all games, we obtain the upper bound on $\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{CHAPO}}(\lambda)$ given in the theorem, which concludes the proof. □

Theorem 3. *If* PKE *is correct and key verifiable, then* CHAPO *is* Integrity*-secure. Moreover, it holds that* $\mathbf{Adv}^{\mathsf{Integrity}}_{\mathsf{CHAPO}}(\lambda) = 0$.

Proof. We prove this theorem by negation. Assume that the adversary returns $(i, j, (c_l, \mathsf{pk}_l)_{i \leq l \leq j}, x, \mathsf{sk}_x, \mathsf{ik}_{i \to j})$ to the challenger such that running $(m_l)_{i \leq l \leq j} \leftarrow \mathsf{Open}(\mathsf{sko}_*, i, j, \mathsf{ik}_{i \to j}, (c_l, \mathsf{pk}_l)_{i \leq l \leq j})$ and $m'_x \leftarrow \mathsf{Dec}(\mathsf{sk}_x, c_x)$, it holds that $m_x \neq \perp$ and $m'_x \neq \perp$ and $m_x \neq m'_x$ and $1 = \mathsf{KVer}(\mathsf{pk}_x, \mathsf{sk}_x)$. We show that this implies the following contradiction: $m'_x = m_x$.

We parse c_x as $(\widetilde{C}_x, \widehat{C}_x, D_x, S_x, T_x, x)$. According to the algorithm Open, if $m_x \neq \perp$, then during the run of $\mathsf{Open}(\mathsf{sko}_*, i, j, \mathsf{ik}_{i \to j}, (c_l, \mathsf{pk}_l)_{i \leq l \leq j})$, the challenger computes two values mk_x and v_x such that $D_x = \mathsf{PEnc}(\mathsf{pk}_x, [m_x \| \mathsf{mk}_x \| x]; v_l)$. We have that $1 = \mathsf{KVer}(\mathsf{pk}_x, \mathsf{sk}_x)$, so $(\mathsf{pk}_x, \mathsf{sk}_x) \in \{(\mathsf{pk}, \mathsf{sk}) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KVer}(1^\lambda)\}$. Moreover, since PKE is correct, then we have that for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KVer}(1^\lambda)$, any message m and any random coin r, it holds that $m = \mathsf{PDec}(\mathsf{sk}, \mathsf{PEnc}(\mathsf{pk}, m; r))$. We deduce that $[m_x \| \mathsf{mk}_x \| x] = \mathsf{PDec}(\mathsf{sk}_x, D_x)$.

On the other hand, According to the algorithm Dec, $m'_x \leftarrow \mathsf{Dec}(\mathsf{sk}_x, c_x)$ implies that there exists a values mk'_x such that $[m'_x \| \mathsf{mk}'_x \| x] = \mathsf{PDec}(\mathsf{sk}_x, D_x)$. We deduce that $m'_x = m_x$, which concludes the proof. □

5 Applications

Chosen ciphertext security in practice. Our CCA security model provides several properties that are essential for practical applications: the IND-CCA security ensures that the ciphertexts are not malleable, and the IND-CSCA ensures that if an adversary modifies, drops, adds or rearranges the encrypted messages in the open interval, the open algorithm fails. In what follows, we show several applications of APOPKE, and we discuss the practical impact of the CCA security.

Encrypted invoices during tax audit. Bultel and Lafourcade give the following application for their APOPKE scheme. A company sends invoices to its customers by encrypted e-mail. One day, the company has to pass a tax audit, and the court asks the company's mail server to provide the invoices sent over

a given period of time. The company does not want to reveal the invoices of its customers that were not sent over the period of the tax audit. It therefore uses an APOPKE to reveal only the invoices sent over the time period. The CPA security only takes into account the cases where the server is passive: a dishonest server could discredit the company in court by adding or removing invoices, or changing their amounts in an undetectable way. The dishonest server can also modify the invoices before the clients receive them. Our CCA security fixes this drawback by preventing this kind of attacks.

Interception of secure messaging and vocal chat. In the introduction, we have already mentioned the use of APOPKE to reveal his textual conversations during a given period of time in a court of law, in order to prove his honesty. Again, a dishonest server might want to alter the user's conversations, so the CCA security is necessary. Often messaging applications also implement voice calls. In this case, encrypting the conversation with a public key encryption becomes infeasible for the sake of effectiveness. However, it is possible to use an APOPKE to encapsulate a session key used throughout the call to encrypts the packets with an authenticated encryption scheme. [22]. To open the conversations over the interval, the opener will decrypt each session key using his interval key, then use each session key to decrypt the packets from the corresponding conversation. From a security point of view, the CCA security prevents the server from altering the session key, and the authenticated encryption ensures the integrity of the conversations decrypted with the session key.

Bypass the limitations. The main limitation of CHAPO is that the user cannot securely generate keys for two disjoint intervals. This limitation can be bypassed in practice by refreshing the user's encryption key from time to time. In this case, the refresh rate must be chosen so as to obtain the best tradeoff between the number of encryption keys and the number of ciphertexts in the period covered by each key. To take the example of the instant messaging, we can consider that the user must refresh his encryption key every year. On the one hand, he is unlikely to accumulate several court cases for different periods during a year, and on the other hand, even if he uses the instant messaging all his life, he will only need to store less than one hundred keys. The user will have to send as many interval keys as there are years in the selected period to the opener, which seems reasonable in practice.

6 Conclusion

In this paper, we revisited the *a posteriori* openable encryption schemes introduced by Bultel and Lafourcade in [5]. We gave a security model for the chosen-ciphertext attack security of this primitive, and we proposed a new scheme called CHAPO which improves four points of the scheme given in [5]: it is more secure (CCA security), more generic, more efficient, and it does not require the random oracle model. We also presented new applications for this primitive. In the future, we would like to put these applications into practice. Moreover, it would be interesting to adapt the *a posteriori* openable encryptions to the secure messaging based on the double ratchet algorithm [7], such as Signal or Watsapp.

Acknowledgements

I would like to thank Angèle Bossuat and David Gérault for their helpful comments and suggestions.

References

1. G. Arfaoui, O. Blazy, X. Bultel, P.-A. Fouque, A. Nedelcu, and C. Onete. Legally keeping secrets from mobile operators: Lawful interception key exchange (like). In *ESORICS 2021 (forthcoming)*, 2021.
2. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT 2000*. Springer, 2000.
3. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO '96*. Springer, 1996.
4. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, 1997.
5. X. Bultel and P. Lafourcade. A posteriori openable public key encryption. In *ICT Systems Security and Privacy Protection*. Springer International Publishing, 2016.
6. G. Choi and S. Vaudenay. Timed-release encryption with master time bound key. In *Information Security Applications*. Springer International Publishing, 2020.

7. K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. In *EuroS&P 2017*, 2017.
8. Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *EUROCRYPT 2002*, LNCS. Springer, 2002.
9. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 1986.
10. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *45th ACM STOC*. ACM Press, 2013.
11. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 1984.
12. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS '06*. ACM, 2006.
13. K. Han, C. Y. Yeun, T. Shon, J. H. Park, and K. Kim. A scalable and efficient key escrow model for lawful interception of idbc-based secure communication. *Int. J. Communication Systems*, 2011.
14. Y. H. Hwang, D. H. Yum, and P. J. Lee. Timed-release encryption with pre-open capability and its application to certified e-mail system. In *ISC 2005*, LNCS. Springer, 2005.
15. M. Ishizaka and S. Kiyomoto. Time-specific encryption with constant-size secret-keys secure under standard assumption. Cryptology ePrint Archive, Report 2020/595, 2020. `https://eprint.iacr.org/2020/595`.
16. K. Kasamatsu, T. Matsuda, K. Emura, N. Attrapadung, G. Hanaoka, and H. Imai. Time-specific encryption from forward-secure encryption: generic and direct constructions. *International Journal of Information Security*, 2015.
17. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS*. ACM, 2013.
18. K. M. Martin. Increasing efficiency of international key escrow in mutually mistrusting domains. In *Cryptography and Coding*, LNCS. Springer, 1997.
19. T. May. Time-release crypto. Manuscript, 1993.
20. R. Nojima, H. Imai, K. Kobara, and K. Morozov. Semantic security for the mceliece cryptosystem without random oracles. *Des. Codes Cryptography*, 2008.
21. K. G. Paterson and E. A. Quaglia. Time-specific encryption. In *SCN'10*, 2010.
22. P. Rogaway. Authenticated-encryption with associated-data. In *CCS*. ACM, 2002.
23. A. Shamir. Partial key escrow: A new approach to software key escrow. Presented at Key Escrow Conference, 1995.
24. Z. Wang, Z. Ma, S. Luo, and H. Gao. Key escrow protocol based on a tripartite authenticated key agreement and threshold cryptography. *IEEE Access*, 2019.
25. Y. Watanabe and J. Shikata. Identity-based hierarchical key-insulated encryption without random oracles. In *PKC 2016*. Springer, 2016.

A Full Security Proofs

In this section, we give the full verion of the proofs, including the reductions.

We prove each theorem by using a sequence of games between a *challenger* and an *adversary* where the adversary plays an experiment run by the challenger. Our proofs use the following lemmas (Lemma 2 is proven in [2]):

Lemma 1. *For any security property* Prop*, any PPT algorithm* $\mathcal{A}$*, and any scheme* P*, if* $\mathbf{Adv}^{\mathsf{Prop}}_{P,\mathcal{A}}(\lambda) = |\mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{b,P,\mathcal{A}}(\lambda)] - 1/2|$*, then it holds that* $|\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{1,P,\mathcal{A}}(\lambda)] - \mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{0,P,\mathcal{A}}(\lambda)]| = 2 \cdot \mathbf{Adv}^{\mathsf{Prop}}_{P,\mathcal{A}}(\lambda)$.

Proof. First, we show that:

$$\begin{aligned}
&\mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{b,P,\mathcal{A}}(\lambda)] \\
&= \mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : b = 1] \cdot \mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{1,P,\mathcal{A}}(\lambda)] \\
&+ \mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : b = 0] \cdot \mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{0,P,\mathcal{A}}(\lambda)] \\
&= \frac{1}{2} \cdot \Big(\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{1,P,\mathcal{A}}(\lambda)] + 1 - \mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{0,P,\mathcal{A}}(\lambda)]\Big) \\
&= \frac{1}{2} \cdot \Big(\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{1,P,\mathcal{A}}(\lambda)] - \mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{0,P,\mathcal{A}}(\lambda)]\Big) + \frac{1}{2}.
\end{aligned}$$

We deduce that:

$$\begin{aligned}
2 \cdot \mathbf{Adv}^{\mathsf{Prop}}_{P,\mathcal{A}}(\lambda) &= 2 \cdot \Big|\mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{b,P,\mathcal{A}}(\lambda)] - 1/2\Big| \\
&= \Big|\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{1,P,\mathcal{A}}(\lambda)] - \mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{Prop}}_{0,P,\mathcal{A}}(\lambda)]\Big|.
\end{aligned}$$

Lemma 2. *Let λ be a security parameter and $P = (\mathcal{R}_\lambda, \mathsf{PGen}, \mathsf{PEnc}, \mathsf{PDec})$ be a PKE, and let $\mathcal{A}$ be a PPT algorithm. The (q_e, μ)-*indistinguishability against chosen ciphertexts attacks security in a multi-users/challenges setting *((q_e, μ)-$\mathsf{IND\text{-}CCA}$) experiment for P is defined by:*
$\mathbf{Exp}^{(q_e,\mu)\text{-}\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)$*:*
$\mathcal{C} \leftarrow \emptyset$;$\forall i \in [\![1, \mu]\!], (\mathsf{pk}_{(*,i)}, \mathsf{sk}_{(*,i)}) \leftarrow \mathsf{PGen}(1^\lambda)$;
$b_* \leftarrow \mathcal{A}^{\mathsf{PEnc}(\cdot,\mathsf{LR}_b(\cdot,\cdot)),\mathsf{PDec}(\cdot,\cdot)}(\lambda, (\mathsf{pk}_i)_{1\leq i\leq \mu})$*; If $b = b_*$, then return 1, else 0;*
where the oracle $\mathsf{PEnc}(\cdot, \mathsf{LR}_b(\cdot,\cdot))$ cannot be called more than q_e times and takes as input $(i, (m_0, m_1))$, picks $r \leftarrow \mathcal{R}_\lambda$, runs $c \leftarrow \mathsf{PEnc}(\mathsf{pk}_i, m_b; r)$, sets $\mathcal{C} \leftarrow \mathcal{C} \cup \{(i, c)\}$, and returns c, and where the oracle $\mathsf{PDec}(\cdot,\cdot)$ takes a pair (i, c) as input, and returns $\mathsf{PDec}(\mathsf{sk}_i, c)$ if $(i, c) \notin \mathcal{C}$. The (q_e, μ)-$\mathsf{IND\text{-}CCA}$-advantage is defined by $\mathbf{Adv}^{(q_e,\mu)\text{-}\mathsf{IND\text{-}CCA}}_{P,\mathcal{A}}(\lambda) = \left|\mathsf{Pr}[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{(q_e,\mu)\text{-}\mathsf{IND\text{-}CCA}}_{b,P,\mathcal{A}}(\lambda)] - 1/2\right|$.
It holds that $\mathbf{Adv}^{(q_e,\mu)\text{-}\mathsf{IND\text{-}CCA}}_{P}(\lambda) \leq q_e \cdot \mu \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{P}(\lambda)$.

A.1 IND-CSCA Security Proof.

Proof. (Theorem 1) Let $\mathcal{A}$ be a PPT algorithm, we use the following sequence of games:
Game G_0: This game is the original $\mathsf{IND\text{-}CSCA}$ experiment:

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_0\right] = \mathsf{Pr}\left[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CSCA}}_{b,\mathsf{CHAPO},\mathcal{A}}(\lambda)\right]$$

Game G_1: This game is similar to G_0, but the challenger replaces each output of the PRF by a random value picked at random in $\mathcal{K}^s_\lambda$. We claim that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_0\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_1\right]| \leq \mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the PR experiment using $\mathcal{A}$ as a black box. $\mathcal{B}$ faithfully simulates G_0 to $\mathcal{A}$, except that it instantiates k_* by picking $\mathsf{mk}_* \xleftarrow{\$} \mathcal{K}^m_\lambda$ and setting $\mathsf{k}_* \leftarrow (\perp, \mathsf{mk}_*)$, and it calls the oracle $f(\cdot)$ instead of running the pseudo random function $\mathsf{PRF}(\mathsf{rk}_*, \cdot)$ on any input ($\mathcal{B}$ cannot run the decryption algorithm since it does not know the real key rk_*). If $\mathcal{A}$ wins the game, then $\mathcal{B}$ returns 1, else it returns 0. We have that:
- $\mathsf{Pr}\left[\mathsf{rk} \xleftarrow{\$} \mathcal{K}_\lambda; f(.) \leftarrow \mathsf{PRF}(\mathsf{rk}, \cdot); : 1 \leftarrow \mathcal{B}^{f(\cdot)}(\lambda)\right] = \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_0\right]$,
- $\mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F} : 1 \leftarrow \mathcal{B}^{f(\cdot)}(\lambda)\right] = \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_1\right]$.

This concludes the proof of the claim.
Game G_2: Let q_x be the number of queries sent to the oracle $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$. This game is similar to G_1, but the challenger replaces the elements encrypted in the interval keys by random values:
- At the beginning of the experiment, the challenger initializes an empty list $\mathcal{L}_X[\,]$.
- Each time that the oracle $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$ receives a query (i, j), the challenger parses k_* as $(\mathsf{rk}_*, \mathsf{mk}_*)$. If $i \leq 0$, or $i \geq j$, or $j \geq n$, or $[\![i, j]\!] \cap [\![i_*, j_*]\!] \neq \emptyset$, then it returns $\perp$ according to the definition of the oracle Ext. It then picks $r \xleftarrow{\$} \mathcal{R}^p_\lambda$ and $\mathsf{rnd} \xleftarrow{\$} \{0,1\}^{|\widetilde{\mathsf{ek}}_i \| \widehat{\mathsf{ek}}_j \| \mathsf{mk}_*|}$, runs $X_{i\to j} \leftarrow \mathsf{PEnc}(\mathsf{pko}_*, \mathsf{rnd}; r)$, sets $\mathcal{L}_{\mathsf{Ext}}[X_{i\to j}] \leftarrow [i \| j]$, runs $Y_{i\to j} \leftarrow \mathsf{Mac}(\mathsf{mk}_*, [i \| j \| X_{i\to j}])$, sets $\mathsf{ik}_{i\to j} \leftarrow (X_{i\to j}, Y_{i\to j}, i, j)$ and returns $\mathsf{ik}_{i\to j}$.
- Each time that the challenger runs $\mathsf{Open}(\mathsf{sko}_*, i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\leq l\leq j})$ such that, parsing $\mathsf{ik}_{i\to j}$ as $(X_{i\to j}, Y_{i\to j}, i, j)$, $\mathcal{L}_X[X_{i\to j}] \neq \perp$, it replaces the instruction $[\widetilde{\mathsf{ek}}'_i \| \widehat{\mathsf{ek}}'_j \| \mathsf{mk}'] \leftarrow \mathsf{PDec}(\mathsf{sko}_*, X_{i\to j})$ by the instructions $[i' \| j'] \leftarrow \mathcal{L}_X[X_{i\to j}]$; $\widetilde{\mathsf{ek}}'_i \leftarrow \widetilde{\mathsf{ek}}_{i'}$; $\widehat{\mathsf{ek}}'_j \leftarrow \widehat{\mathsf{ek}}_{j'}$; $\mathsf{mk}' \leftarrow \mathsf{mk}_*$.

We claim that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_1\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_2\right]| \leq 2 \cdot q_x \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the $(q_x, 1)$-$\mathsf{IND\text{-}CCA}$ experiment on PKE using $\mathcal{A}$ as a black box. $\mathcal{B}$ receives the public key (pk'_*), then it simulates honestly the game G_2 to $\mathcal{A}$, except that:
- $\mathcal{B}$ sets $\mathsf{pko}_* \leftarrow \mathsf{pk}'_*$ and $\mathsf{sko}_* \leftarrow \perp$.
- Each time that the oracle $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$ receives a query (i, j), it parses k_* as $(\mathsf{rk}_*, \mathsf{mk}_*)$. If $i \leq 0$, or $i \geq j$, or $j \geq n$, or $[\![i, j]\!] \cap [\![i_*, j_*]\!] \neq \emptyset$, then it returns $\perp$, else it picks $\mathsf{rnd} \xleftarrow{\$} \{0,1\}^{|\widetilde{\mathsf{ek}}_i \| \widehat{\mathsf{ek}}_j \| \mathsf{mk}_*|}$. It build the query $q = (1, ([\widetilde{\mathsf{ek}}_i \| \widehat{\mathsf{ek}}_j \| \mathsf{mk}_*], \mathsf{rnd}))$ and sends it to its encryption oracle $\mathsf{PEnc}(\cdot, \mathsf{LR}_{b'}(\cdot, \cdot))$, that retunrs a ciphertext c, then $\mathcal{B}$ sets $X_{i\to j} \leftarrow c$ and $\mathcal{L}_X[X_{i\to j}] \leftarrow [i \| j]$, runs $Y_{i\to j} \leftarrow \mathsf{Mac}(\mathsf{mk}_*, [i \| j \| X_{i\to j}])$, sets $\mathsf{ik}_{i\to j} \leftarrow (X_{i\to j}, Y_{i\to j}, i, j)$ and returns $\mathsf{ik}_{i\to j}$.

– Each time that the challenger runs $\mathsf{Open}(\mathsf{sko}_*, i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\le l\le j})$, parsing $\mathsf{ik}_{i\to j}$ as $(X_{i\to j}, Y_{i\to j}, i, j)$:
  - if $\mathcal{L}_X[X_{i\to j}] \neq \perp$, then the challenger replaces the instruction $[\widetilde{\mathsf{ek}}_i'\|\widehat{\mathsf{ek}}_j'\|\mathsf{mk}'] \leftarrow \mathsf{PDec}(\mathsf{sko}_*, X_{i\to j})$ by the instructions $[i'\|j'] \leftarrow \mathcal{L}_X[X_{i\to j}]$; $\widetilde{\mathsf{ek}}_i' \leftarrow \widetilde{\mathsf{ek}}_{i'}$; $\widehat{\mathsf{ek}}_j' \leftarrow \widehat{\mathsf{ek}}_{j'}$; $\mathsf{mk}' \leftarrow \mathsf{mk}_*$.
  - if $\mathcal{L}_X[X_{i\to j}] = \perp$, then instead of running the instruction $[\widetilde{\mathsf{ek}}_i'\|\widehat{\mathsf{ek}}_j'\|\mathsf{mk}'] \leftarrow \mathsf{PDec}(\mathsf{sko}_*, X_{i\to j})$ ($\mathcal{B}$ cannot run the PDec algorithm since it does not know the key sko_*), $\mathcal{B}$ sends the query $(1, X_{i\to j})$ to the oracle $\mathsf{PDec}(\cdot,\cdot)$ and parses the answer as $[\widetilde{\mathsf{ek}}_i'\|\widehat{\mathsf{ek}}_j'\|\mathsf{mk}']$.

At then end of the game simulation, $\mathcal{A}$ returns b_*, then $\mathcal{B}$ returns 1 to its challenger iff $(b = b_*)$.

In what follows, b' denotes the challenge bit of $\mathcal{B}$. If $b' = 0$, then the challenger encrypts $[\widetilde{\mathsf{ek}}_i\|\widehat{\mathsf{ek}}_j\|\mathsf{mk}_*]$ in c, so $\mathsf{ik}_{i\to j} = c$ is formed as in the game G_1, and the oracle Open is perfectly simulated because the instruction $[\widetilde{\mathsf{ek}}_i'\|\widehat{\mathsf{ek}}_j'\|\mathsf{mk}'] \leftarrow \mathsf{PDec}(\mathsf{sko}_*, X_{i\to j})$ and the instructions $[i'\|j'] \leftarrow \mathcal{L}_X[X_{i\to j}]$; $\widetilde{\mathsf{ek}}_i' \leftarrow \widetilde{\mathsf{ek}}_{i'}$; $\widehat{\mathsf{ek}}_j' \leftarrow \widehat{\mathsf{ek}}_{j'}$; $\mathsf{mk}' \leftarrow \mathsf{mk}_*$ are equivalent. Else, the challenger encrypts a random bit-string in c, so $\mathsf{ik}_{i\to j} = c$ is formed as in the game G_2.

We deduce that $\mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{(q_x,1)\text{-IND-CCA}}_{0,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}((\mathsf{pk}_*'), \lambda)|b' = 0] = \mathsf{Pr}[b_* = b|b' = 0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]$ and $\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{(q_x,1)\text{-IND-CCA}}_{1,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}((\mathsf{pk}_*'), \lambda)|b' = 1] = \mathsf{Pr}[b_* = b|b' = 1] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_2]$. By Lemma 1, we have that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_2]| \le 2 \cdot \mathbf{Adv}^{(q_x,1)\text{-IND-CCA}}_{\mathsf{SKE},\mathcal{B}}(\lambda)$. By Lemma 2, we have that $\mathbf{Adv}^{(q_x,1)\text{-IND-CCA}}_{\mathsf{SKE}}(\lambda) \le q_x \cdot \mathbf{Adv}^{\mathsf{IND-CCA}}_{\mathsf{SKE}}(\lambda)$, which concludes the proof of the claim.

We stress that from this game, **the key mk_* is never used by the challenger except to generate/verify the message authentication codes**.

Game G_3: In this game, the challenger aborts and returns a random bit if the adversary sends a valid query with a fresh interval key $\mathsf{ik}_{i\to j}$ to the oracle Open, *i.e.* an interval key that has not been generated by the experiment and that does not abort the oracle. more precisely, this game is similar to G_2, but:

– At the beginning, the challenger initializes an empty set $\mathcal{S}_{\mathsf{ik}}$.
– Each time that the challenger generates an interval key $\mathsf{ik}_{i\to j}$, the challenger sets $\mathcal{S}_{\mathsf{ik}} \leftarrow \mathcal{S}_{\mathsf{ik}} \cup \{\mathsf{ik}_{i\to j}\}$.
– Each time that the oracle $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ receives a query $(i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\le l\le j})$ such that the oracle does not returns $\perp$, if parsing $\mathsf{ik}_{i\to j}$ as $(X_{i\to i}, Y_{i\to i}, i, j)$ it holds that $\mathsf{ik}_{i\to j} \notin \mathcal{S}_{\mathsf{ik}}$ and $1 = \mathsf{Ver}(\mathsf{mk}_*, Y_{i\to j}, [i\|j\|X_{i\to i}])$, then the challenger sets $\mathsf{Abort}_3 \leftarrow 1$, aborts the experiment and returns a random bit.

We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_3]| \le \mathbf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{MAC}}(\lambda).$$

We have that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_3]| \le \mathsf{Pr}[\mathsf{Abort}_3 = 1]$. We prove this claim by showing that $\mathsf{Pr}[\mathsf{Abort}_3 = 1] \le \mathbf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{MAC}}(\lambda)$ using a reduction. Assume that the adversary $\mathcal{A}$ sends a query to the oracle Open that makes the experiment aborting with probability $\epsilon_{\mathcal{A}}(\lambda)$, *i.e.* $\mathsf{Pr}[\mathsf{Abort}_3 = 1] = \epsilon_{\mathcal{A}}(\lambda)$. We build an algorithm $\mathcal{B}$ that plays the $\mathsf{EUF\text{-}CMA}$ experiment on MAC using $\mathcal{A}$ as a black box. $\mathcal{B}$ simulates G_3 to $\mathcal{A}$ as in the real game, except that:

– $\mathcal{B}$ sets $\mathsf{mk}_* \leftarrow \perp$.
– If $\mathcal{B}$ has to run $t \leftarrow \mathsf{Mac}(\mathsf{mk}_*, m)$ on some input m, then it sends m to the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$ and instantiates t with the answer of the oracle.
– If $\mathcal{B}$ has to check that $1 = \mathsf{Mac}(\mathsf{mk}_*, t, m)$ for some input (t, m), then it sends (t, m) to the oracle $\mathsf{Ver}(\mathsf{mk}_*, \cdot, \cdot)$.
– If $\mathcal{A}$ sends a query to the oracle $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ such that:
  - the oracle do not return $\perp$, and
  - parsing $\mathsf{ik}_{i\to j} = (X_{i\to i}, Y_{i\to i}i, j)$, it holds that $\mathsf{ik}_{i\to j} \notin \mathcal{S}_{\mathsf{ik}}$ and $1 = \mathsf{Ver}(\mathsf{mk}_*, Y_{i\to j}, [i\|j\|X_{i\to i}])$,

  then $\mathcal{B}$ returns $(Y_{i\to j}, [i\|j\|X_{i\to i}])$ to its challenger, sets $\mathsf{Abort}_3 \leftarrow 1$, and aborts the experiment for $\mathcal{A}$. In this case, the (valid) pair of tag/message $(Y_{i\to j}, [i\|j\|X_{i\to i}])$ as not been produced by the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$. We deduce that $\mathcal{B}$ wins his $\mathsf{EUF\text{-}CMA}$ experiment.

Finally, we observe that $\mathcal{B}$ perfectly simulates G_3 to $\mathcal{A}$, and if Abort_3 triggers, then $\mathcal{B}$ wins its $\mathsf{EUF\text{-}CMA}$ experiment. We deduce that $\epsilon_{\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{MAC},\mathcal{B}}(\lambda)$, which concludes the proof of the claim.

Game G_4: In this game, the challenger aborts and returns a random bit if the adversary sends a valid query with a fresh ciphertext c_l to the oracle Open, *i.e.* a ciphertext that has not been generated by the experiment and that does not abort the oracle. more precisely, this game is similar to G_3, but:

– At the beginning of the experiment, the challenger initializes an empty set $\mathcal{S}_{\mathsf{Enc}}$.
– Each time that the challenger runs the encryption algorithm $c \leftarrow \mathsf{Enc}(\mathsf{pk}_{(*,q)}, \mathsf{k}_*, m, l; r)$ for any input m and l, and any key index q, the challenger sets $\mathcal{S}_{\mathsf{Enc}} \leftarrow \mathcal{S}_{\mathsf{Enc}} \cup \{(\mathsf{pk}_{(*,q)}, c)\}$.
– Each time that the oracle $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ receives a query $(i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\leq l\leq j})$ such that $\exists\ l$ such that parsing c_l as $(\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$, it holds that $(\mathsf{pk}_l, c_l) \notin \mathcal{S}_{\mathsf{Enc}}$ and $1 = \mathsf{Ver}(\mathsf{mk}_*, S_l, [\widetilde{C}_l \| \widehat{C}_l \| D_l \| \mathsf{pk}_l \| l])$, and such that the oracle do not return $\perp$, the challenger sets $\mathsf{Abort}_4 \leftarrow 1$, aborts the experiment and returns a random bit.

We claim that:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_4\right]| \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

We have that $|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_3\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_4\right]| \leq \mathsf{Pr}[\mathsf{Abort}_4 = 1]$. We prove this claim by showing that $\mathsf{Pr}[\mathsf{Abort}_4 = 1] \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda)$ using a reduction. Assume that the adversary $\mathcal{A}$ sends a query to the oracle Open that makes the experiment aborting with probability $\epsilon_{\mathcal{A}}(\lambda)$, *i.e.* $\mathsf{Pr}[\mathsf{Abort}_4 = 1] = \epsilon_{\mathcal{A}}(\lambda)$. We build an algorithm $\mathcal{B}$ that plays the $\mathsf{EUF\text{-}CMA}$ experiment on MAC using $\mathcal{A}$ as a black box. $\mathcal{B}$ simulates G_4 to $\mathcal{A}$ as in the real game, except that:

– $\mathcal{B}$ sets $\mathsf{mk}_* \leftarrow \perp$.
– If $\mathcal{B}$ has to run $t \leftarrow \mathsf{Mac}(\mathsf{mk}_*, m)$ on some input m, then it sends m to the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$ and instantiates t with the answer of the oracle.
– If $\mathcal{B}$ has to check that $1 = \mathsf{Mac}(\mathsf{mk}_*, t, m)$ for some input (t, m), it sends (t, m) to the oracle $\mathsf{Ver}(\mathsf{mk}_*, \cdot, \cdot)$.
– If $\mathcal{A}$ sends a query $(i, j, \mathsf{ik}_{i\to j}, (c_l, \mathsf{pk}_l)_{i\leq l\leq j})$ to the oracle Open such that $\exists\ l$ such that parsing c_l as $(\widetilde{C}_l, \widehat{C}_l, D_l, S_l, T_l, l)$ it holds that $(\mathsf{pk}_l, c_l) \notin \mathcal{S}_{\mathsf{Enc}}$ and $1 = \mathsf{Ver}(\mathsf{mk}_*, S_l, [\widetilde{C}_l \| \widehat{C}_l \| D_l \| \mathsf{pk}_l \| l])$, then we distinguish two cases:
  • If $\exists\ (\mathsf{pk}, c) \in \mathcal{S}_{\mathsf{Enc}}$ such that parsing c as $(\widetilde{C}, \widehat{C}, D, S, T, l')$ it holds that $(\widetilde{C}, \widehat{C}, D, S, l') = (\widetilde{C}_l, \widehat{C}_l, D_l, S_l, l)$ and $T \neq T_l$ and $\mathsf{pk}_l = \mathsf{pk}$, then $\mathcal{B}$ aborts the oracle and returns $\perp$.
  We note that in this case, the games G_3 and G_4 are correctly simulated: in the real games, when it runs the algorithm Open, the challenger computes the values m_l, mk_l and v_l, and checks that $D_l = \mathsf{PEnc}(\mathsf{pk}_l, [m_l \| \mathsf{mk}_l \| l]; v_l)$ and $T_l = \mathsf{Mac}(\mathsf{mk}_l, [\widetilde{C}_l \| \widehat{C}_l \| D_l \| S_l \| \mathsf{pk}_l \| l])$, otherwise it returns $\perp$ by definition of the algorithm Open. Since PKE is correct, there does not exist any (m', v') such that $m' \neq [m_l \| \mathsf{mk}_l \| l]$ and $D_l = \mathsf{PEnc}(\mathsf{pk}_l, m'; v')$, so if the algorithm Open does not abort, then mk_l is fixed by D_l and cannot be altered. Moreover, Mac is a deterministic algorithm, which implies that if $(\widetilde{C}, \widehat{C}, D, S, l') = (\widetilde{C}_l, \widehat{C}_l, D_l, S_l, l)$ and $T_l \neq T$, then $T_l \neq \mathsf{Mac}(\mathsf{mk}_l, [\widetilde{C}_l \| \widehat{C}_l \| D_l \| S_l \| \mathsf{pk}_l \| l])$, so the algorithm Open should return $\perp$ in this case.
  • Else, $\mathcal{B}$ returns $(S_l, [\widetilde{C}_l \| \widehat{C}_l \| D_l \| \mathsf{pk}_l \| l])$ to its challenger, it sets $\mathsf{Abort}_4 \leftarrow 1$, and it aborts the experiment for $\mathcal{A}$.
  In this case, since $\forall\ \{(\mathsf{pk}, c)\} \in \mathcal{S}_{\mathsf{Enc}}$ such that $\mathsf{pk}_l = \mathsf{pk}$, and parsing c as $(\widetilde{C}, \widehat{C}, D, S, l')$, it holds that $(\widetilde{C}, \widehat{C}, D, l') \neq (\widetilde{C}_l, \widehat{C}_l, D_l, l)$, c has not be produced by $\mathcal{B}$, which implies that the (valid) pair of tag/message $(S_l, [\widetilde{C}_l \| \widehat{C}_l \| D_l \| \mathsf{pk}_l \| l])$ as not been produced by the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$. We deduce that in this case, $\mathcal{B}$ wins his $\mathsf{EUF\text{-}CMA}$ experiment.

Finally, $\mathcal{B}$ perfectly simulates G_4 to $\mathcal{A}$, and if Abort_4 triggers, then $\mathcal{B}$ wins its $\mathsf{EUF\text{-}CMA}$ experiment. We deduce that $\epsilon_{\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC},\mathcal{B}}(\lambda)$, which concludes the proof of the claim.

We stress that from this game:

– **The oracle Open aborts if it receives a query that contains pairs of public key/ciphertext that has not been created by the challenger, or if the key $\mathsf{ik}_{i\to j}$ has not been created by the challenger**.
– Moreover, since the index is a part of the ciphertext, **the oracle Open aborts if it receives a query that contains a vector of public keys/ciphertexts which are not ordered by their encryption order**.
– Since the index of the first and the last ciphertexts are parts of the interval key $\mathsf{ik}_{i\to j}$, **the oracle aborts if it receives an interval key that has not been created for matching the ciphertext interval**.
– Finally, since the experiment never produce an interval key for an interval (i, j) such that $[\![i, j]\!] \cap [\![i_*, j_*]\!] \neq \emptyset$ except for $\mathsf{ik}_{i_*\to j_*}$, and since the oracle aborts on the query $(\mathsf{ik}_{i_*\to j_*}, (c_{(*,l)}, \overline{\mathsf{pk}}_l)_{i_*\leq l\leq j_*})$, then **the oracle Open aborts and returns $\perp$ on any interval (i, j) such that it holds that $[\![i, j]\!] \cap [\![i_*, j_*]\!] \neq \emptyset$.**

Game G_5: In what follows, c_l denotes the encryption of the l^{th} message m_l by the experiment, and $\bar{\mathsf{pk}}_l$ denotes the corresponding public key. Moreover, for any pair of indexes (i,j), $\mathsf{ik}^h_{i\to j}$ denotes the h^{th} interval key returned by the oracle $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$ on the input (i,j). This game is the same as G_5 but the oracle $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ is re-defined by:

- On input $(i, j, \mathsf{ik}'_{i\to j}, (c'_l, \mathsf{pk}'_l)_{i\le l\le j})$, if $i \le 0$ or $i \ge j$ or $j \ge n$, then return $\perp$. During the second phase, if $(\mathsf{ik}'_{i\to j}, (c'_l, \mathsf{pk}'_l)_{i\le l\le j}) = (\mathsf{ik}_{i_*\to j_*}, (c_{(*,l)}, \bar{\mathsf{pk}}_l)_{i_*\le l\le j_*})$, then return $\perp$.
- If $(c'_l, \mathsf{pk}'_l)_{i\le l\le j} = (c_l, \mathsf{pk}_l)_{i\le l\le j}$ and $\exists\ h$ such that $\mathsf{ik}'_{i\to j} = \mathsf{ik}^h_{i\to j}$, then return $(m_l)_{i\le l\le j}$, else return $\perp$.

From the properties of G_4 given above, we deduce that:

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_4\right] = \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_5\right].$$

We stress that from this game, **the challenger no longer use the algorithm** SDec **during the experiment**.

Game G_6: In this game, the challenger replaces the message encrypted in each ciphertext $\widetilde{C}_l$ by a random value. More precisely, this game is similar to G_5, but each time that the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ is run by the Enc oracle or by the experiment during the generation of the challenge, the challenger replaces the instruction $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]; \widetilde{u}_n)$ by the sequence of instructions $\mathsf{str}_n \leftarrow [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, \mathsf{rnd}_n; \widetilde{u}_n)$;

Let q_* be an upper bound on the number of ciphertexts sending by the adversary $\mathcal{A}_1$, *i.e.* $j_* - i_* \le q_*$. We claim that:

$$\left|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_5\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_6\right]\right| \le 2 \cdot (q_n + q_*) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

We prove this claim by using an hybrid argument. We define an hybrid game $G_{6,h}$ as follows:

Game $G_{6,h}$: In this game, the challenger replaces the messages encrypted in the h first ciphertexts $\widetilde{C}_l$ by random values. More precisely, If $h = 0$, then $G_{6,h} = G_5$, else if $1 \le h \le (q_n + q_*)$, then the game $G_{6,h}$ is the same as $G_{6,h-1}$, but when the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ is run for $n = h$ by the Enc oracle or by the experiment during the generation of the challenge, the challenger replaces the instructions $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]; \widetilde{u}_n)$ by $\mathsf{str}_n \leftarrow [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, \mathsf{rnd}_n; \widetilde{u}_n)$.

We claim that, for all $h \in [\![1, q_n + q_*]\!]$:

$$\left|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{6,h-1}\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{6,h}\right]\right| \le 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the $\mathsf{IND\text{-}CCA}$ experiment on SKE using $\mathcal{A}$ as a black box. $\mathcal{B}$ simulates honestly the game $G_{6,h-1}$ to $\mathcal{A}$, except that:

- It sets $\widetilde{ek}_h \leftarrow \perp$.
- If $\mathcal{B}$ runs $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ such that $n = h$ (by simulating the Enc oracle or by generating the challenge), then $\mathcal{B}$ replaces the instruction $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]; \widetilde{u}_n)$ by running $\mathsf{str}_n \leftarrow [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $(\bar{m}_0, \bar{m}_1) \leftarrow (\mathsf{str}_n, \mathsf{rnd}_n)$; then by sending $(\bar{m}_0, \bar{m}_1)$ to its challenger and by instantiating $\widetilde{C}_n$ with the ciphertext c_* returned by the challenger.

At the end of the game simulation, $\mathcal{A}$ returns b_*, then $\mathcal{B}$ returns 1 to its challenger iff $(b = b_*)$.

In what follows, b' denotes the challenge bit of the $\mathsf{IND\text{-}CCA}$ experiment on SKE played by $\mathcal{B}$. We first note that in $G_{6,h-1}$, the ciphertext $\widetilde{C}_{h-1}$ encrypts a random message, which means that the key $\widetilde{ek}_h$ is never used in the game $G_{6,h}$, except for encrypting $\widetilde{C}_h$. If $(b' = 0)$, then the $\mathsf{IND\text{-}CCA}$ challenger of $\mathcal{B}$ encrypts $\bar{m}_0 = \mathsf{str}_h$, so $G_{6,h-1}$ is perfectly simulated by $\mathcal{B}$, else, $(b' = 1)$ and $\widetilde{C}_h$ encrypts the random value $\bar{m}_1 = \mathsf{rnd}_h$, so $G_{6,h}$ is perfectly simulated. We deduce that $\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{1,\mathsf{SKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda) | b' = 1] = \mathsf{Pr}[b_* = b | b' = 1] = \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{6,h}\right]$. On the other hand, $\mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{0,\mathsf{SKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda) | b' = 0] = \mathsf{Pr}[b_* = b | b' = 0] = \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{6,h-1}\right]$. By Lemma 1, we have that $\left|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{6,h-1}\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{6,h}\right]\right| = 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE},\mathcal{B}}(\lambda)$, which concludes the proof of the claim. Moreover, we have that $G_{6,q_n+q_*} = G_6$, which concludes the proof of the claim of the Game 6.

Game G_7: In this game, the challenger replaces the message encrypted in each ciphertext $\widehat{C}_l$ by a random value. More precisely, this game is similar to G_6, but when the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*,$

$m_n, n; r_n)$ is run for $n = h$ by the Enc oracle or by the experiment during the generation of the challenge, the challenger replaces the instructions $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]; \widehat{u}_n)$ by the sequence $\mathsf{str}_n \leftarrow [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, \mathsf{rnd}_n; \widehat{u}_n)$. Let q_* be an uper bound on the number of ciphertexts sending by the adversary $\mathcal{A}_1$, *i.e.* $j_* - i_* \leq q_*$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_6] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]| \leq 2 \cdot (q_n + q_*) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

This claim can be proved using the following hybrid argument: We define an hybrid game $G_{7,h}$ as follows:

Game $G_{7,h}$: In this game, the challenger replaces the messages encrypted in the $(q_n + q_* + 1 - h)$ last ciphertexts $\widetilde{C}_l$ by random values. More precisely, if $h = q_n + q_* + 1$, then $G_{7,h} = G_6$, else if $1 \leq h \leq (q_n + q_*)$, then the game $G_{7,h}$ is the same as $G_{7,h+1}$, but each time the challenger runs the encryption algorithm $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$ during the Enc oracle or during the challenge generation such that $n = h$, it replaces the instruction $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]; \widehat{u}_n)$ by the sequence $\mathsf{str}_n \leftarrow [\widehat{\mathsf{ek}}_{n-1} \| \widehat{m}_n \| \widehat{\mathsf{mk}}_n \| \widehat{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, \mathsf{rnd}_n; \widehat{u}_n)$. We claim that, for all $h \in [\![1, q_n + q_*]\!]$:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{7,h+1}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{7,h}]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

This claim can be proven by reduction. We omit the details of this proof because it is very similar to the proof in the game $G_{6,h}$.

Game G_8: In what follows, we parse the l^{th} ciphertext $c_{(*,l)}$ of the challenge by $(\widetilde{C}_{(*,l)}, \widehat{C}_{(*,l)}, D_{(*,l)}, S_{(*,l)}, T_{(*,l)}, l)$. In this game, the challenger replaces the keys mk_l for all l in $[\![i_*, j_*]\!]$ encrypted by a public key generated by the challenger (*i.e.* in $\{\mathsf{pk}_{(*,q)}\}_{1 \leq q \leq \mu}$) in each ciphertext $D_{(*,l)}$ by a random value. More precisely, this game is similar to G_7, except that when the challenger build the challenge $(\mathsf{ik}_{i_* \to j_*}, (c_{(*,l)})_{i_* \leq l \leq j_*})$, $\forall\ l \in [\![i_*, j_*]\!]$ such that $\exists\ q, \mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$:

- the challenger picks $\mathsf{mk}'_l \xleftarrow{\$} \mathcal{K}^m_\lambda$,
- when the challenger runs $c_{(*,l)} \leftarrow \mathsf{Enc}(\bar{\mathsf{pk}}_l, \mathsf{k}_*, m_{(b,l)}, l)$, it replaces the instruction $D_{(*,l)} \leftarrow \mathsf{PEnc}(\mathsf{pk}_{(*,q)}, [m_{(b,l)} \| \mathsf{mk}_l \| l]; v_l)$ by $D_{(*,l)} \leftarrow \mathsf{PEnc}(\mathsf{pk}_{(*,q)}, [m_{(b,l)} \| \mathsf{mk}'_l \| l]; v_l)$.
- For each query (p, c) (where we parse c as $(\widetilde{C}, \widehat{C}, D, S, T, l')$) sending to the oracle $\mathsf{Dec}(\cdot, \cdot)$ such that $p = q$, $c \neq c_{(*,l)}$ and $D = D_{(*,l)}$, the challenger runs $\mathsf{Dec}(\mathsf{sk}_{(*,q)}, c)$ as in the real experiment except that it replaces the instruction $[m \| \mathsf{mk} \| l''] \leftarrow \mathsf{PDec}(\mathsf{sk}_{(*,q)}, D)$ by $[m \| \mathsf{mk} \| l''] \leftarrow [m_{(b,l)} \| \mathsf{mk}_l \| l]$.

We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_7] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_8]| \leq 2 \cdot q_* \cdot \mu \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the (q_*, μ)-IND-CCA experiment on PKE using $\mathcal{A}$ as a black box. $\mathcal{B}$ receives a public key set $(\mathsf{pk}_{(*,l)})_{1 \leq l \leq \mu}$, then it runs $\mathcal{A}_1((\mathsf{pk}_{(*,l)})_{1 \leq l \leq \mu}, \lambda)$. It simulates honestly the game G_7 to $\mathcal{A}$, except that $\forall\ l \in [\![1, \mu]\!]$ it sets $\mathsf{sk}_{(*,l)} = \perp$, and $\forall\ l \in [\![i_*, j_*]\!]$ such that $\exists\ q, \mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$:

- $\mathcal{B}$ picks $\mathsf{mk}'_l \xleftarrow{\$} \mathcal{K}^m_\lambda$,
- when $\mathcal{B}$ runs $c_{(*,l)} \leftarrow \mathsf{Enc}(\bar{\mathsf{pk}}_l, \mathsf{k}_*, m_{(b,l)}, l)$ during the challenge phase, it replaces the instruction $D_{(*,l)} \leftarrow \mathsf{PEnc}(\mathsf{pk}_{(*,q)}, [m_{(b,l)} \| \mathsf{mk}_l \| l]; v_l)$ by the following instructions: $\mathcal{B}$ picks $\mathsf{mk}'_l \xleftarrow{\$} \mathcal{K}^m_\lambda$, then it sends $(q, [m_{(b,l)} \| \mathsf{mk}_l | l], [m_{(b,l)} \| \mathsf{mk}'_l | l])$ to its oracle $\mathsf{PEnc}(\cdot, \mathsf{LR}_b(\cdot, \cdot))$, and it instantiates $D_{(*,l)}$ by the value returned by $\mathsf{PEnc}(\cdot, \mathsf{LR}_b(\cdot, \cdot))$.
- For each query (p, c) (where we parse c as $(\widetilde{C}, \widehat{C}, D, S, T, l')$) sending by $\mathcal{A}$ to the oracle $\mathsf{Dec}(\cdot, \cdot)$ such that $c \neq c_{(*,l)}$ and $p = q$, $\mathcal{B}$ runs $\mathsf{Dec}(\mathsf{sk}_{(*,q)}, c)$ as in the real experiment except that if $D = D_{(*,l)}$, it replaces the instruction $[m \| \mathsf{mk} \| l''] \leftarrow \mathsf{PDec}(\mathsf{sk}_{(*,q)}, D_{(*,l)})$ by $[m \| \mathsf{mk} \| l''] \leftarrow [m_{(b,l)} \| \mathsf{mk}_l \| l]$. Else, $\mathcal{B}$ sends (q, D) to the oracle $\mathsf{PDec}(\cdot, \cdot)$ and parses the output of the oracle as $[m \| \mathsf{mk} \| l'']$.

Finally, $\mathcal{A}_2$ returns b_* and $\mathcal{B}$ returns 1 iff $(b = b_*)$.

Let b' be the challenge bit of the IND-CCA experiment on PKE played by $\mathcal{B}$. If $b' = 0$, then $D_{(*,l)}$ encrypts $[m_{(b,l)} \| \mathsf{mk}_l \| l]$, so G_7 is perfectly simulated to $\mathcal{A}$. We deduce that $\mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{(q_*,\mu)\text{-IND-CCA}}_{0,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}(\lambda, (\mathsf{pk}_{(*,q)})_{1 \leq q \leq \mu}) | b' = 0] = \mathsf{Pr}[b_* = b | b' = 0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]$. On the other hand, if $b' = 1$, then $D_{(*,l)}$ encrypts $[m_{(b,l)} \| \mathsf{mk}'_l \| l]$, so G_8 is perfectly simulated to $\mathcal{A}$. In this case, we have that $\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{(q_*,\mu)\text{-IND-CCA}}_{1,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}(\lambda, (\mathsf{pk}_{(*,q)})_{1 \leq q \leq \mu}) | b' = 1] = \mathsf{Pr}[b_* = b | b' = 1] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_8]$. By

Lemma 1 and Lemma 2, we deduce that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_7] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_8]| = 2 \cdot \mathbf{Adv}^{(q_*,\mu)\text{-IND-CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda) \leq 2 \cdot q_* \cdot \mu \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda)$, which concludes the proof of the claim.

We stress that from this game, for each index l in $[\![i_*, j_*]\!]$ such that $\bar{\mathsf{pk}}_l \in \{\mathsf{pk}_{(*,q)}\}_{1\leq q\leq \mu}$, **the key mk_l is only used to produced the MAC tag $T_{(*,l)}$. Especially, mk_l is never encrypted in a ciphertext known by the adversary**.

Game G_9: In this game, the challenger tries to guess the indexes i_* and j_*, and aborts in case of failure. More precisely, this game is similar to G_8, but the challenger picks $(i'_*, j'_*) \leftarrow [\![1, q_n + q_*]\!]^2$ at the beginning of the experiment. At the end of the experiment, if $i'_* \neq j_*$ or $i'_* \neq j_*$, then the challenger returns a random bit. The adversary increases its winning advantage by a factor equalling the probability of guessing correctly i_* and j_*:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_8] - 1/2| = (q_n + q_*)^2 \cdot |\mathsf{Pr}[\mathcal{A} \text{ wins } G_9] - 1/2|$$

We stress that n is incremented $(q_n + q_*)$ times during the experiment: q_n times by the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$ and q_* times after the generation of $(c_{(*,l)})_{i_*\leq l\leq j_*}$.

Game G_{10}: We parse the challenge $(c_{(*,l)})_{i_*\leq l\leq j_*}$ as $(\widetilde{C}_{(*,l)}, \widehat{C}_{(*,l)}, D_{(*,l)}, S_{(*,l)}, T_{(*,l)}, l)_{i_*\leq l\leq j_*}$. In this game, the challenger aborts if the adversary tries to reuse the element $D_{(*,l)}$ in a ciphertext $c \neq c_{(*,l)}$ sending to the decryption oracle for all l in $[\![i_*, j_*]\!]$ such that $\bar{\mathsf{pk}}_l \in \{\mathsf{pk}_{(*,q)}\}_{1\leq q\leq \mu}$. More concretely, this game is similar to G_9, but if the adversary sends a query (p, c) (where we parse c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$) to the oracle $\mathsf{Dec}(\cdot,\cdot)$ such that, $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$ and $D = D_{(*,l)}$ and $c \neq c_{(*,l)}$ and $\mathsf{Ver}(\mathsf{mk}_l, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,q)}\|l]) = 1$, then the challenger set $\mathsf{Abort}_{10} \leftarrow 1$, aborts the game G_{10} and returns a random bit. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_9] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{10}]| \leq q_* \cdot \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

We prove this claim by using an hybrid argument. We define the game $G_{10,h}$ as follows:

Game $G_{10,h}$: We define $G_{10,0}$ as G_9, and for all $1 \leq h \leq q_*$, we define $G_{10,h}$ as $G_{10,h-1}$ but the challenger aborts if (i) $\bar{\mathsf{pk}}_{(*,i'_*+h)} \in \{\mathsf{pk}_{(*,q)}\}_{1\leq q\leq \mu}$, and (ii) the adversary tries to reuse the element $D_{(*,i'_*+h)}$ in a ciphertext $c \neq c_{(*,i'_*+h)}$ sending to the decryption oracle. More concretely, we define $G_{10,h}$ as the same game as $G_{10,h-1}$ except that parsing the challenge $(c_{(*,l)})_{i_*\leq l\leq j_*}$ as $(\widetilde{C}_{(*,l)}, \widehat{C}_{(*,l)}, D_{(*,l)}, S_{(*,l)}, T_{(*,l)})_{i_*\leq l\leq j_*}$, if the adversary sends a query (p, c) to the oracle $\mathsf{Dec}(\cdot,\cdot)$ such that, parsing c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$, it holds that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_{i'_*+h}$ and $D = D_{(*,i'_*+h)}$ and $c \neq c_{(*,i'_*+h)}$ and $\mathsf{Ver}(\mathsf{mk}_{i'_*+h}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,q)}\|l]) = 1$, then the challenger set $\mathsf{Abort}_{10,h} \leftarrow 1$, aborts the game $G_{10,h}$ and returns a random bit. We note that $G_{10,q_*} = G_{10}$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{10,h-1}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{10,h}]| \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

We have that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{10,h-1}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{10,h}]| \leq \mathsf{Pr}[\mathsf{Abort}_{10,h} = 1]$. To prove this claim, we show that $\mathsf{Pr}[\mathsf{Abort}_{10,h} = 1] \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda)$. Assume that the adversary $\mathcal{A}$ sends a query (p, c) to the oracle $\mathsf{Dec}(\cdot,\cdot)$ such that, parsing c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$, it holds that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_{i'_*+h}$ and $D = D_{(*,i'_*+h)}$ and $c \neq c_{(*,i'_*+h)}$ and $\mathsf{Ver}(\mathsf{mk}_{i'_*+h}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,q)}\|l]) = 1$ with probability $\epsilon_{\mathcal{A}}(\lambda)$, *i.e.* $\mathsf{Pr}[\mathsf{Abort}_{10,h} = 1] = \epsilon_{\mathcal{A}}(\lambda)$. We build the following algorithm $\mathcal{B}$ that plays the MAC experiment using $\mathcal{A}$ as a black box.

$\mathcal{B}$ simulates $G_{10,h}$ to $\mathcal{A}$ as in the real game, except that:

- $\mathcal{B}$ sets $\mathsf{mk}_{i'_*+h} \leftarrow \perp$.
- In what follows, we parse the $(i'_* + h)^{\text{th}}$ ciphertext of the challenge $c_{(*,i'_*+h)}$ as $(\widetilde{C}_{(*,i'_*+h)}, \widehat{C}_{(*,i'_*+h)}, D_{(*,i'_*+h)}, S_{(*,i'_*+h)}, T_{(*,i'_*+h)}, i'_* + h)$. When $\mathcal{B}$ runs $c_{(*,i'_*+h)} \leftarrow \mathsf{Enc}(\bar{pk}_{i'_*+h}, \mathsf{k}_*, m_{(b,i'_*+h)}, i'_* + h; r_{(*,i'_*+h)})$, instead of running $T_{(*,i'_*+h)} \leftarrow \mathsf{Mac}(\mathsf{mk}_{i'_*+h}, [\widetilde{C}_{(*,i'_*+h)}\|\widehat{C}_{(*,i'_*+h)}\|D_{(*,i'_*+h)}\|S_{(*,i'_*+h)}\|\bar{\mathsf{pk}}_{i'_*+h}\|i'_* + h])$ ($\mathcal{B}$ cannot run the Mac algorithm since it does not know the real key $\mathsf{mk}_{i'_*+h}$), it sends $[\widetilde{C}_{(*,i'_*+h)}\|\widehat{C}_{(*,i'_*+h)}\|D_{(*,i'_*+h)}\|S_{(*,i'_*+h)}\|\bar{\mathsf{pk}}_{i'_*+h}\|i'_* + h]$ to the oracle $\mathsf{Mac}(\mathsf{mk}_*,\cdot)$ which retuns a tag t, then $\mathcal{B}$ sets $T_{(*,i'_*+h)} \leftarrow t$.
- If $\mathcal{A}$ sends a query (p, c) to the oracle $\mathsf{Dec}(\cdot,\cdot)$ such that, parsing c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$, it holds that $\bar{\mathsf{pk}}_{i'_*+h} = \mathsf{pk}_{(*,p)}$ and $D = D_{(*,i'_*+h)}$ and $c \neq c_{(*,i'_*+h)}$ and the oracle $\mathsf{Ver}(\mathsf{mk}_*,\cdot,\cdot)$ returns 1 on $(T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,p)}\|l])$, then $\mathcal{B}$ sets $\mathsf{Abort}_{10,h} = 1$, aborts $G_{10,h}$, and returns $(T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,p)}\|$

$l]$). We remark that if $c \neq c_{(*,i'_*+h)}$, then $T \neq T_{(*,i'_*+h)}$ or $(\widetilde{C}, \widehat{C}, D, S, l) \neq (\widetilde{C}_{(*,i'_*+h)}, \widehat{C}_{(*,i'_*+h)}, D_{(*,i'_*+h)}, S_{(*,i'_*+h)}, i'_* + h)$, which implies that $(T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,p)}\|l]) \neq (T_{(*,i'_*+h)}, [\widetilde{C}_{(*,i'_*+h)}\|\widehat{C}_{(*,i'_*+h)}\|D_{(*,i'_*+h)}\|S_{(*,i'_*+h)}\|\bar{\mathsf{pk}}_{i'_*+h}\|i'_* + h])$.

We observe that that if $\mathsf{Abort}_{10,h} = 1$, then $\mathcal{B}$ wins the EUF-CMA experiment since it returns a fresh pair of tag/message, *i.e.* a pair that has not been outputted by the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$. Hence, it holds that $\mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B},\mathsf{MAC}}(\lambda) = \epsilon_{\mathcal{A}}(\lambda)$, which concludes the proof of this claim, and therefor the proof of the claim in G_{10}.

We stress that from this step, if the adversary sends a query (p, c) to the oracle $\mathsf{Dec}(\cdot,\cdot)$ such that, parsing c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$, there exists l' in $[\![i_*, j_*]\!]$ such that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_{l'}$ and $D = D_{(*,l')}$, then:

- If $l \neq l'$, then the oracle returns $\perp$: indeed, this oracle runs the algorithm Dec, which runs $\bar{m} \leftarrow \mathsf{PDec}(\mathsf{sk}_{(*,q)}, D_{(*,l')})$. According to how $D_{(*,l')}$ has been encrypted by the challenger, it holds that $\bar{m} = [m_{(b,l')}\|\mathsf{mk}_{l'}\|l']$, and Dec returns $\perp$ if $l \neq l'$.
- If $c = c_{(*,l')}$, then the oracle returns $\perp$ by definition of the oracle $\mathsf{Dec}(\cdot,\cdot)$.
- If $\mathsf{Ver}(\mathsf{mk}_{l'}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,q)}\|l]) \neq 1$, then the oracle returns $\perp$ by definition of the algorithm Dec.
- If $c \neq c_{(*,l')}$ and $\mathsf{Ver}(\mathsf{mk}_{l'}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,q)}\|l]) = 1$, then the experiment aborts by definition of G_{10}

In any case, **if $\exists l' \in [\![i_*, j_*]\!]$ such that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_{l'}$ and $D = D_{(*,l')}$, then the oracle does not return $\mathsf{Dec}(\mathsf{sk}_{(*,q)}, c)$ to the challenger, which implies that the adversary can no longer use the decryption oracle to decrypt the ciphertexts of the challenge that depend on b.**

Game G_{11}: This game is similar to G_{10}, but the challenger substitutes each message that depends on b by a random message of same length. More formally, for each $l \in [\![i'_*, j'_*]\!]$ such that $\bar{\mathsf{pk}}_l \in \{\mathsf{pk}_{(*,q)}\}_{1\leq q\leq \mu}$ it replaces the message $m_{(b,l)}$ encrypted in $c_{(*,l)}$ by the public key $\bar{\mathsf{pk}}_l$ by a message $m_{(*,l)} \xleftarrow{\$} \{0,1\}^{|m_{(b,l)}|}$ chosen at random. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{10}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{11}]| = 2 \cdot q_* \cdot \mu \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the (q_*, μ)-IND-CCA experiment on PKE using $\mathcal{A}$ as a black box. $\mathcal{B}$ receives a set of public key $(\mathsf{pk}_{(*,l)})_{1\leq l\leq \mu}$, then it runs $\mathcal{A}_1((\mathsf{pk}_{(*,l)})_{1\leq l\leq \mu}, \lambda)$. It simulates honestly the game G_{10} to $\mathcal{A}$, except that $\forall l \in [\![i'_*, j'_*]\!]$ it sets $\mathsf{sk}_{(*,l)} = \perp$ and:

- In what follows, we parse the each ciphertext $c_{(*,l)}$ of the challenge as $(\widetilde{C}_{(*,l)}, \widehat{C}_{(*,l)}, D_{(*,l)}, S_{(*,l)}, T_{(*,l)}, l)$. $\forall l \in [\![i'_*, j'_*]\!]$ such that $\exists q$ such that $\bar{\mathsf{pk}}_l = \mathsf{pk}_{(*,q)}$, when $\mathcal{B}$ runs $c_{(*,l)} \leftarrow \mathsf{Enc}(\bar{\mathsf{pk}}_l, \mathsf{k}_*, m_{(b,l)}, l; r_l)$, instead of running $D_{(*,l)} \leftarrow \mathsf{PEnc}(\mathsf{pk}_{(*,q)}, [m_{(b,l)}\|\mathsf{mk}'_l\|l]; v_l)$, it picks $m_{(*,l)} \xleftarrow{\$} \{0,1\}^{|m_{(b,l)}|}$, then it sends $(q, ([m_{(b,l)}\|\mathsf{mk}'_l\|l], [m_{(*,l)}\|\mathsf{mk}'_l\|l]))$ to the oracle $\mathsf{PEnc}(\cdot, \mathsf{LR}_b(\cdot,\cdot))$ and instantiates $D_{(*,l)}$ by the answer.
- For each query (q, c) sending by $\mathcal{A}$ to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$, we parse c as $(\widetilde{C}, \widehat{C}, D, S, T, l)$, and:
  - If $\exists l \in [\![i_*, j_*]\!]$ such that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$ and $D = D_{(*,l)}$ and $\mathsf{Ver}(\mathsf{mk}_l, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,q)}\|l]) \neq 1$, then $\mathcal{B}$ aborts the game according to the algorithm Dec.
  - If $\exists l \in [\![i_*, j_*]\!]$ such that $\mathsf{pk}_{(*,q)} = \bar{\mathsf{pk}}_l$ and $D = D_{(*,l)}$ and $\mathsf{Ver}(\mathsf{mk}_l, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_{(*,q)}\|l]) = 1$, then $\mathcal{B}$ aborts the game according to the condition introduced in G_{10}.
  - Else , $\mathcal{B}$ runs $\mathsf{Dec}(\mathsf{sk}_{(*,q)}, c)$ as in the real experiment except that instead of running $[m\|\mathsf{mk}\|l] \leftarrow \mathsf{PDec}(\mathsf{sk}_{(*,q)}, D)$ ($\mathcal{B}$ cannot run the decryption algorithm since it does not know the real key $\mathsf{sk}_{(*,q)}$), it sends D to the oracle $\mathsf{PDec}(\mathsf{sk}_{(*,q)}, \cdot)$ and parses the answer of the oracle as $[m\|\mathsf{mk}\|l]$.
- $\mathcal{B}$ simulates $\mathsf{Ext}(\mathsf{pko}_*, \mathsf{k}_*, \cdot, \cdot)$ and $\mathsf{Open}(\mathsf{sko}_*, \cdot, \cdot, \cdot, \cdot)$ as in the real game G_{10}. Concerning Open, we recall that $\mathcal{B}$ never answers to the queries that contain some pairs of public key/ciphertext or some key $\mathsf{ik}_{i\to j}$ that has not been created by the challenger, and never answers to the queries concerning some interval (i, j) such that $[\![i, j]\!] \cap [\![i_*, j_*]\!] \neq \emptyset$. We deduce that $\mathcal{B}$ never answers to the queries that contain any ciphertext $c_{(*,l)}$ for $l \in [\![i_*, j_*]\!]$. All other kind of queries can be answered by $\mathcal{B}$ as in the real game G_{10}.

Finally, $\mathcal{A}_2$ returns b_* and $\mathcal{B}$ returns 1 iff $(b = b_*)$.

Let b' be the challenge bit of the (q_*, μ)-IND-CCA experiment. If $b' = 0$, then each $D_{(*,l)}$ encrypts $[m_{(b,l)}\|\mathsf{mk}'_l\|l]$, so G_{10} is perfectly simulated to $\mathcal{A}$. We deduce that $\mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{(q_*,\mu)\text{-IND-CCA}}_{0,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[b_* = b | b' = 0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_{10}]$. On the other hand, if $b' = 1$, then $D_{(*,l)}$ encrypts $[m_{(*,l)}\|\mathsf{mk}'_l\|l]$, so G_{11} is perfectly simulated to $\mathcal{A}$. In this case, we have that $\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{(q_*,\mu)\text{-IND-CCA}}_{1,\mathsf{PKE},\mathcal{B}}(\lambda)] =$

$\mathsf{Pr}[b_* = b | b' = 1] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_{11}]$. By Lemma 1 and Lemma 2, we deduce that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{10}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{11}]| = 2 \cdot \mathbf{Adv}^{(q_*,\mu)\text{-IND-CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda) \leq 2 \cdot q_* \cdot \mu \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda)$, which concludes the proof of the claim.
Conclusion: Finally, since the game G_{11} do not depend on the challenge bit b, we have that $\mathsf{Pr}[\mathcal{A} \text{ wins } G_{11}] = 1/2$. By composing the winning probabilities of $\mathcal{A}$ in all games, we obtain the upper bound on $\mathbf{Adv}^{\mathsf{IND\text{-}CSCA}}_{\mathsf{CHAPO}}(\lambda)$ given in the theorem, which concludes the proof. □

A.2 IND-CCA Security Proof.

Proof. (Theorem 2) We recall that in order to win the IND-CCA experiment with a non negligible advantage, the following must hold: $l_* \notin [\![i_*, j_*]\!]$. We separate our proof in two distinct cases: $l_* < i_*$ and $l_* > j_*$. More concretely, we define two variants of the IND-CCA experiment: the $\mathsf{IND\text{-}CCA}_0$ (resp. $\mathsf{IND\text{-}CCA}_1$) experiment denotes the same experiment as IND-CCA except that if $l_* < i_*$ (resp. $l_* > j_*$), then the challenger returns a random bit. We have that:

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{CHAPO},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{IND\text{-}CCA}_0}_{\mathsf{CHAPO},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{IND\text{-}CCA}_1}_{\mathsf{CHAPO},\mathcal{A}}(\lambda).$$

Case $\mathsf{IND\text{-}CCA}_0$ (*i.e.* $l_* < i_*$): We use the following sequence of games. Let $\mathcal{A}$ be a PPT algorithm:

Game G_0: This game is the original $\mathsf{IND\text{-}CCA}_0$ experiment:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] = \mathsf{Pr}\left[b \xleftarrow{\$} \{0,1\} : 1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}_0}_{b,\mathsf{CHAPO},\mathcal{A}}(\lambda)\right]$$

Game G_1: This game is similar to G_0, but the challenger replaces each output of the PRF by a random value picked in $\mathcal{K}^s_\lambda$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_0] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_1]| \leq \mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF}}(\lambda).$$

We prove this claim by reduction in the same way as in the game G_1 of the proof of Theorem 1.
Game G_2: In this game, the challenger tries to guess the index i_* and aborts in case of failure. Let q_n be the number of calls to the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$. This game is similar to G_1, but the challenger picks $i'_* \leftarrow [\![1, q_n + 1]\!]$ at the beginning of the experiment. At the end of the experiment, if $i'_* \neq i_*$, then the challenger returns a random bit. The adversary increases its winning advantage by a factor equalling the probability of guessing correctly i_*:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_1] - 1/2| = (q_n + 1) \cdot |\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] - 1/2|$$

Note that n is incremented $(q_n + 1)$ times during the experiment: q_n times by the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$ and one time after the generation of c_{l_*}.
Game G_3: In this game, for all $n \leq i_*$, the challenger replaces the part $\widetilde{C}_n$ of the ciphertext c_n by the encryption of a random message, which includes the challenge c_{l_*} if $l_* \leq i_*$ (otherwise, the challenger returns a random bit by definition of the $\mathsf{IND\text{-}CCA}_0$ experiment). This game is similar to G_2, but while $n < i'_*$, each time the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$ is called on a query (pk_n, m_n) and runs $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$, it replaces the instruction $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]; \widetilde{u}_n)$ by the sequence of instructions $\mathsf{str}_n \leftarrow [\widetilde{\mathsf{ek}}_{n+1} \| \widetilde{m}_n \| \widetilde{\mathsf{mk}}_n \| \widetilde{v}_n \| \mathsf{pk}_n \| n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widetilde{C}_n \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_n, \mathsf{rnd}_n; \widetilde{u}_n)$. Moreover, when it computes the challenge $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,0)}, l_*; r_*)$, the challenger replaces the instruction $\widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, [\widetilde{\mathsf{ek}}_{l_*+1} \| \widetilde{m}_{(*,b)} \| \widetilde{\mathsf{mk}}_{l_*} \| \widetilde{v}_{l_*} \| \mathsf{pk}_* \| l_*]; \widetilde{u}_{l_*})$; by the sequence $\mathsf{str}_{l_*} \leftarrow [\widetilde{\mathsf{ek}}_{l_*+1} \| \widetilde{m}_{(*,b)} \| \widetilde{\mathsf{mk}}_{l_*} \| \widetilde{v}_{l_*} \| \mathsf{pk}_* \| l_*]$; $\mathsf{rnd}_{l_*} \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_{l_*}|}$; $\widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, \mathsf{rnd}_{l_*}; \widetilde{u}_{l_*})$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_3]| \leq 2 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

We prove this claim by using an hybrid argument. We define the hybrid game $G_{3,i}$ as follows:
Game $G_{3,i}$: If $i = 0$, then $G_{3,0} = G_2$, else for all $n \leq i$, if $i \leq i'_*$, then the challenger replaces the part $\widetilde{C}_n$ of the ciphertext c_n by the encryption of a random message. More concretely, if $1 \leq i \leq (q_n + 1)$, then the game $G_{3,i}$ is the same as $G_{3,i-1}$, but if $i < i'_*$ and $l_* \neq i$, then when the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, i)$ is called on a query (pk_i, m_i) and runs $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{k}_*, m_i, i; r_i)$, it replaces the instruction $\widetilde{C}_i \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_i,$

$[\widetilde{\mathsf{ek}}_{i+1}\|\widetilde{m}_i\|\widetilde{\mathsf{mk}}_i\|\widetilde{v}_i\|\mathsf{pk}_i\|i];\widetilde{u}_i)$ by the sequence of instructions $\mathsf{str}_i[\widetilde{\mathsf{ek}}_{i+1}\|\widetilde{m}_i\|\widetilde{\mathsf{mk}}_i\|\widetilde{v}_i\|\mathsf{pk}_i\|i];\ \mathsf{rnd}_i \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_i|};\ \widetilde{C}_i \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_i, \mathsf{rnd}_i; \widetilde{u}_i)$. Moreover, if $l_* = i$, then when it computes the challenge $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,0)}, l_*; r_*)$, the challenger replaces the instruction $\widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, [\widetilde{\mathsf{ek}}_{l_*+1}\|\widetilde{m}_{(*,b)}\|\widetilde{\mathsf{mk}}_{l_*}\|\widetilde{v}_{l_*}\|\mathsf{pk}_*\|l_*]; \widetilde{u}_{l_*})$ by $\mathsf{str}_{l_*} \leftarrow [\widetilde{\mathsf{ek}}_{l_*+1}\|\widetilde{m}_{(*,b)}\|\widetilde{\mathsf{mk}}_{l_*}\|\widetilde{v}_{l_*}\|\mathsf{pk}_*\|l_*];\ \mathsf{rnd}_{l_*} \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_{l_*}|};\ \widetilde{C}_{l_*} \leftarrow \mathsf{SEnc}(\widetilde{\mathsf{ek}}_{l_*}, \mathsf{rnd}_{l_*}; \widetilde{u}_{l_*})$. We claim that, for all $i \in [\![1, q_n+1]\!]$:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{3,i-1}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{3,i}]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the IND-CCA experiment on SKE using $\mathcal{A}$ as a black box. $\mathcal{B}$ simulates honestly the game $G_{3,i-1}$ to $\mathcal{A}$, except that:

- If $i < i'_*$ and $l_* \neq i$, then when the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, i)$ is called on a query (pk_i, m_i), $\mathcal{B}$ runs $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{k}_*, m_i, i; r_i)$ as in the definition of CHAPO, except that it sets $\widetilde{\mathsf{ek}}_i \leftarrow \perp$, sets $\bar{m}_0 \leftarrow [\widetilde{\mathsf{ek}}_{i+1}\|\widetilde{m}_i\|\widetilde{\mathsf{mk}}_i\|\widetilde{v}_i\|\mathsf{pk}_i\|i]$, picks $\bar{m}_1 \xleftarrow{\$} \{0,1\}^{|\bar{m}_0|}$, sends $(\bar{m}_0, \bar{m}_1)$ to its challenger, receives the challenge c_* and uses it to instantiate $\widetilde{C}_i$, *i.e.* $\widetilde{C}_i \leftarrow c_*$.
- If $i < i'_*$ and $l_* = i$, then $\mathcal{B}$ runs $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,b)}, l_*; r_{l_*})$ as in the real game, except that it sets $\widetilde{\mathsf{ek}}_{l_*} \leftarrow \perp$, sets $\bar{m}_0 \leftarrow [\widetilde{\mathsf{ek}}_{l_*+1}\|\widetilde{m}_{(*,b)}\|\widetilde{\mathsf{mk}}_{l_*}\|\widetilde{v}_{l_*}\|\mathsf{pk}_*\|l_*]$, picks $\bar{m}_1 \xleftarrow{\$} \{0,1\}^{|\bar{m}_0|}$, sends $(\bar{m}_0, \bar{m}_1)$ to its challenger, receives the challenge c_* and uses it to instantiate $\widetilde{C}_{l_*}$, *i.e.* $\widetilde{C}_{l_*} \leftarrow c_*$.

At then end of the game simulation, $\mathcal{A}$ returns b_*, then $\mathcal{B}$ returns 1 to its challenger iff $(b = b_*)$.

We first note that in $G_{3,i-1}$, if $i-1 < i < i'_*$, then $\widetilde{C}_{i-1}$ encrypts a random message, which means that the key $\widetilde{ek}_i$ is never used in the experiment, except for encrypting $\widetilde{C}_i$. Moreover, if the IND-CCA challenger of $\mathcal{B}$ encrypts $\bar{m}_0$, then $G_{3,i-1}$ is perfectly simulated by $\mathcal{B}$, else, $\widetilde{C}_i$ encrypts the random value $\bar{m}_1$, so $G_{3,i}$ is perfectly simulated. Let b' be the challenge bit of the IND-CCA experiment of $\mathcal{B}$, we deduce that $\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{1,\mathsf{SKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda, \mathsf{st}, c_*)|b' = 1] = \mathsf{Pr}[b_* = b|b' = 1] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_{3,i}]$. On the other hand, $\mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{0,\mathsf{SKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda, \mathsf{st}, c_*)|b' = 0] = \mathsf{Pr}[b_* = b|b' = 0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_{3,i-1}]$. By Lemma 2, $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_{3,i-1}] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_{3,i}]| = 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE},\mathcal{B}}(\lambda)$, which concludes the proof of the claim. Moreover, we have that $G_{3,q_n+1} = G_3$, which concludes the proof of the claim of the Game 3.

Game G_4: This is the same game as G_3, but $\widehat{m}_{(*,b)}$, $\widehat{\mathsf{mk}}_{l_*}$ and $\widehat{v}_{l_*}$ are substituted by random bit-strings of same length. At this step, since $\widetilde{m}_{(*,b)}$ is replaced by a random bit string in $\widetilde{C}_{l_*}$ and $\widehat{m}_{(*,b)} = m_{(*,b)} \oplus \widetilde{m}_{(*,b)}$, then $\widehat{m}_{(*,b)}$ is indistinguishable from a random bit string, so it no longer depends on b, and can be substituted by a random bit-string without any influence on the adversary advantage. Using a similar argument, we have that $\widehat{\mathsf{mk}}_{l_*}$ and $\widehat{v}_{l_*}$ can also be substituted by random bit-strings. We deduce that:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } G_3] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_4].$$

Game G_5: In this game, the challenger replaces the MAC key mk_{l_*} by a random value in the part D_{l_*} (encrypted by PKE) of the ciphertext challenge c_{l_*}. More concretely, This game is similar to G_5, except that:

- when it computes the challenge by running $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,b)}, l_*; r_*)$, it replaces the instruction $D_{l_*} \leftarrow \mathsf{PEnc}(\mathsf{pk}_*, [m_{(*,b)}\|\mathsf{mk}_{l_*}\|l_*]; v_{l_*})$ by the sequence of instructions $\mathsf{mk}'_{l_*} \xleftarrow{\$} \mathcal{K}^m_\lambda; D_{l_*} \leftarrow \mathsf{PEnc}(\mathsf{pk}_*, [m_{(*,b)}\|\mathsf{mk}'_{l_*}\|l_*]; v_{l_*})$,
- For each query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ sending to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ such that $c \neq c_{l_*}$ and $D = D_{l_*}$, the challenger runs $\mathsf{Dec}(\mathsf{sk}_*, c)$ as in the real experiment except that it replaces the instruction $[m\|\mathsf{mk}\|l] \leftarrow \mathsf{PDec}(\mathsf{sk}_*, D)$ by $[m\|\mathsf{mk}\|l] \leftarrow [m_{(*,b)}\|\mathsf{mk}_{l_*}\|l_*]$.

We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_4] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_5]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the IND-CCA experiment on PKE using $\mathcal{A}$ as a black box. $\mathcal{B}$ receives a public key pk_*, then it runs $\mathcal{A}_1(\lambda, \mathsf{pk}_*)$, and simulates honestly the game G_5 to $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, except that it sets $\mathsf{sk}_* = \perp$ and:

- when it computes the challenge by running $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,b)}, l_*; r_*)$, instead of running $D_{l_*} \leftarrow \mathsf{PEnc}(\mathsf{pk}_*, [m_{(*,b)}\|\mathsf{mk}_{l_*}\|l_*]; v_{l_*})$, it sets $\mathsf{mk}'_{l_*} \xleftarrow{\$} \mathcal{K}^m_\lambda$ and sends $([m_{(*,b)}\|\mathsf{mk}_{l_*}|l_*], [m_{(*,b)}\|\mathsf{mk}'_{l_*}\|l_*])$ to its challenger, which returns the challenge c_*, then $\mathcal{B}$ sets $D_{l_*} \leftarrow c_*$.
- For each query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ sending by $\mathcal{A}$ to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ such that $c \neq c_{l_*}$:
  - if $D = D_{l_*}$, then $\mathcal{B}$ runs $\mathsf{Dec}(\mathsf{sk}_*, c)$ as in the real experiment except that it replaces the instruction $[m\|\mathsf{mk}\|l] \leftarrow \mathsf{PDec}(\mathsf{sk}_*, D)$ by $[m\|\mathsf{mk}\|l] \leftarrow [m_{(*,b)}\|\mathsf{mk}_{l_*}\|l_*]$.

- else, $\mathcal{B}$ runs $\mathsf{Dec}(\mathsf{sk}_*, c)$ as in the real experiment except that instead of running $[m\|\mathsf{mk}\|l] \leftarrow \mathsf{PDec}(\mathsf{sk}_*, D)$ ($\mathcal{B}$ cannot run the decryption algorithm since it does not know the real key sk_*), it sends D to the oracle $\mathsf{PDec}(\mathsf{sk}_*, \cdot)$ and parses the output of the oracle as $[m\|\mathsf{mk}\|l]$.

Finally, $\mathcal{A}_2$ returns b_* and $\mathcal{B}$ returns 1 iff $\mathcal{B}$ did not abort the game and $(b = b_*)$.

Let b' be the challenge bit of the IND-CCA experiment. If $b' = 0$, then D_{l_*} encrypts $[m_{(*,0)}\|\mathsf{mk}_{l_*}\| l_*]$, so G_5 is perfectly simulated to $\mathcal{A}$. We deduce that $\mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{0,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda, \mathsf{pk}_*, \mathsf{st}, c_*)|b' = 0] = \mathsf{Pr}[b_* = b|b' = 0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_4]$. On the other hand, if $b' = 1$, then D_{l_*} encrypts $[m_{(*,0)}\|\mathsf{mk}'_{l_*}\|l_*]$, so G_6 is perfectly simulated to $\mathcal{A}$. In this case, we have that $\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{1,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda, \mathsf{pk}_*\mathsf{st}, c_*)|b' = 1] = \mathsf{Pr}[b_* = b|b' = 1] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_5]$. By Lemma 2, we deduce that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_4] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_5]| = 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$, which concludes the proof of the claim.

Game G_6: In what follows, we parse the challenge c_{l_*} as $(\widetilde{C}_{l_*}, \widehat{C}_{l_*}, D_{l_*}, S_{l_*}, T_{l_*})$. In this game, the challenger aborts if the adversary tries to reuse D_{l_*} in a ciphertext $c \neq c_{l_*}$ sending to the decryption oracle. More concretely, this game is similar to G_5, but if the adversary sends a query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ such that $D = D_{l_*}$, $c \neq c_{l_*}$ and $\mathsf{Ver}(\mathsf{mk}_{l_*}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l]) = 1$, then the challenger set $\mathsf{Abort}_6 \leftarrow 1$, aborts the game G_6 and returns a random bit. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_5] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_6]| \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).$$

We have that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_5] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_6]| \leq \mathsf{Pr}[\mathsf{Abort}_6 = 1]$. Hence, We prove this claim by showing that $\mathsf{Pr}[\mathsf{Abort}_6 = 1] \leq \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda)$ using a reduction. Assume that the adversary $\mathcal{A}$ sends a query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ during the game G_6 such that $D = D_{l_*}$, $c \neq c_{l_*}$ and $\mathsf{Ver}(\mathsf{mk}_{l_*}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l]) = 1$ with probability $\epsilon_{\mathcal{A}}(\lambda)$, *i.e.* $\mathsf{Pr}[\mathsf{Abort}_6 = 1] = \epsilon_{\mathcal{A}}(\lambda)$. We build an algorithm $\mathcal{B}$ that plays the MAC experiment using $\mathcal{A}$ as a black box.

$\mathcal{B}$ simulates G_6 to $\mathcal{A}$ as in the real game, except that:

- When $\mathcal{B}$ computes the challenge by running $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,b)}, l_*; r_*)$, instead of running $T_{l_*} \leftarrow \mathsf{Mac}(\mathsf{mk}_{l_*}, [\widetilde{C}_{l_*}\|\widehat{C}_{l_*}\|D_{l_*}\|S_{l_*}\|\mathsf{pk}_*\|l_*])$ ($\mathcal{B}$ cannot run the MAC algorithm since it does not know the real key mk_{l_*}), it sets $\mathsf{mk}_{l_*} \leftarrow \perp$ and sends $[\widetilde{C}_{l_*}\|\widehat{C}_{l_*}\|D_{l_*}\|S_{l_*}\|\mathsf{pk}_*\|l_*]$ to the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$ which retuns a tag t, then $\mathcal{B}$ sets $T_{l_*} \leftarrow t$.
- If $\mathcal{A}$ sends a query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ during the game G_7 such that $D = D_{l_*}$, $c \neq c_{l_*}$ and the oracle $\mathsf{Ver}(\mathsf{mk}_*, \cdot, \cdot)$ returns 1 on $(T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l])$, then $\mathcal{B}$ sets $\mathsf{Abort}_6 = 1$, aborts G_6, and returns $(T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l])$ to its challenger. We remark that if $c \neq c_{l_*}$, then $T \neq T_{l_*}$ or $(\widetilde{C}, \widehat{C}, D, S, l) \neq (\widetilde{C}_{l_*}, \widehat{C}_{l_*}, D_{l_*}, S_{l_*}, l_*)$, which implies that:

$$(T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l]) \neq (T_{l_*}, [\widetilde{C}_{l_*}\|\widehat{C}_{l_*}\|D_{l_*}\|S_{l_*}\|\mathsf{pk}_*\|l_*]).$$

We observe that if $\mathsf{Abort}_6 = 1$, then $\mathcal{B}$ wins the EUF-CMA experiment since it returns a fresh pair of tag/message, *i.e.* a pair that has not been outputted by the oracle $\mathsf{Mac}(\mathsf{mk}_*, \cdot)$. This concludes the proof of the claim.

We stress that at this step, if the adversary sends a query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ to the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ such that $D = D_{l_*}$, then:

- If $c = c_{l_*}$, then the oracle returns $\perp$ by definition of the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$ in the IND-CCA experiment of APOPKE.
- If $\mathsf{Ver}(\mathsf{mk}_{l_*}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l_*]) \neq 1$, then the oracle returns $\perp$ by definition of the algorithm Dec.
- If $c \neq c_{l_*}$ and $\mathsf{Ver}(\mathsf{mk}_{l_*}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l_*]) = 1$, then the experiment aborts by definition of G_6

In any case, if $D = D_{l_*}$ then the oracle does not return $\mathsf{Dec}(\mathsf{sk}_*, c)$ to the challenger.

Game G_7: This game is similar to G_6, but the challenger substitutes the message $m_{(*,b)}$ by a random message $m_* \xleftarrow{\$} \{0,1\}^{|m_{(*,b)}|}$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_6] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]| = 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda).$$

We prove this claim by reduction. We build an algorithm $\mathcal{B}$ that plays the IND-CCA experiment on PKE using $\mathcal{A}$ as a black box. $\mathcal{B}$ receives a public key pk_*, then it runs $\mathcal{A}_1(\lambda, \mathsf{pk}_*)$, and simulates honestly the game G_6 to $\mathcal{A}$, except that it sets $\mathsf{sk}_* = \perp$ and:

- When $\mathcal{B}$ computes the challenge by running $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,b)}, l_*; r_*)$, instead of running $D_{l_*} \leftarrow \mathsf{PEnc}(\mathsf{pk}_*, [m_{(*,b)}\|\mathsf{mk}'_{l_*}\|l_*]; v_{l_*})$, it picks $m_* \xleftarrow{\$} \{0,1\}^{|m_{(*,b)}|}$ and sends $([m_{(*,b)}\|\mathsf{mk}'_{l_*}\|l_*], [m_*\| \mathsf{mk}'_{l_*}\|l_*])$ to its challenger, which returns the challenge c_*, then $\mathcal{B}$ sets $D_{l_*} \leftarrow c_*$.

– For each query $c = (\widetilde{C}, \widehat{C}, D, S, T, l)$ sending by $\mathcal{A}$ to the decryption oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$:
  - if $D = D_{l_*}$ and $\mathsf{Ver}(\mathsf{mk}_{l_*}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l_*]) = 1$, then $\mathcal{B}$ aborts G_7 according to the condition introduced in G_6.
  - if $D = D_{l_*}$ and $\mathsf{Ver}(\mathsf{mk}_{l_*}, T, [\widetilde{C}\|\widehat{C}\|D\|S\|\mathsf{pk}_*\|l_*]) \neq 1$, then $\mathcal{B}$ returns $\perp$, according to the definition of the oracle $\mathsf{Dec}(\mathsf{sk}_*, \cdot)$.
  - if $D \neq D_{l_*}$, $\mathcal{B}$ runs $\mathsf{Dec}(\mathsf{sk}_*, c)$ as in the real experiment except that instead of running $[m\|\mathsf{mk}\|l] \leftarrow \mathsf{PDec}(\mathsf{sk}_*, D)$ ($\mathcal{B}$ cannot run the decryption algorithm since it does not know the real key sk_*), it sends D to the oracle $\mathsf{PDec}(\mathsf{sk}_*, \cdot)$ and parses the output of the oracle as $[m\|\mathsf{mk}\|l]$.

Finally, $\mathcal{A}_2$ returns b_* and $\mathcal{B}$ returns 1 iff $(b = b_*)$.

Let b' be the challenge bit of the IND-CCA experiment on PKE played by $\mathcal{B}$. If $b' = 0$, then D_{l_*} encrypts $[m_{(*,b)}\|\mathsf{mk}'_{l_*}\|l_*]$, so G_7 is perfectly simulated to $\mathcal{A}$. We deduce that $\mathsf{Pr}[0 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{0,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda, \mathsf{pk}_*, \mathsf{st}, c_*)|b' = 0] = \mathsf{Pr}[b_* = b|b' = 0] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_6]$. On the other hand, if $b' = 1$, then D_{l_*} encrypts $[m_*\|\mathsf{mk}'_{l_*}\|l_*]$, so G_7 is perfectly simulated to $\mathcal{A}$. In this case, we have that $\mathsf{Pr}[1 \leftarrow \mathbf{Exp}^{\mathsf{IND\text{-}CCA}}_{1,\mathsf{PKE},\mathcal{B}}(\lambda)] = \mathsf{Pr}[1 \leftarrow \mathcal{B}_2(\lambda, \mathsf{pk}_*, \mathsf{st}, c_*)|b' = 1] = \mathsf{Pr}[b_* = b|b' = 1] = \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]$. By Lemma 2, we deduce that $|\mathsf{Pr}[\mathcal{A} \text{ wins } G_6] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]| = 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$, which concludes the proof of the claim.

Conclusion of the case IND-CCA$_1$: At this step, the parts $\widehat{C}_{l_*}$, $\widetilde{C}_{l_*}$, and D_{l_*} of the challenge c_{l_*} encrypts random values instead of the messages $\widehat{m}_{(*,b)}$, $\widetilde{m}_{(*,b)}$, and $m_{(*,b)}$, which implies that the game G_7 do not depend on the challenge bit b. We deduce that $\mathsf{Pr}[\mathcal{A} \text{ wins } G_7] = 1/2$. By composing the winning probabilities of $\mathcal{A}$ in all games, we have that:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{IND\text{-}CCA}_0}_{\mathsf{CHAPO}}(\lambda) \leq & \mathbf{Adv}^{\mathsf{PR}}_{\mathsf{PRF}}(\lambda) + 2 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda) \\ & + 4 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(\lambda) + (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\lambda).\end{aligned}$$

Case IND-CCA$_1$ ($l_* > j_*$): We use a similar sequence of games, but G_0, G_2, G_3, and G_4 are replaced by the games G'_0, G'_2, G'_3, and G'_4, such that:

Game G'_0: This game is the original IND-CCA$_1$ experiment.

Game G'_2: This game is similar to G_2, but the challenger picks $j'_* \leftarrow [\![1, q_n + 1]\!]$ at the beginning of the experiment and returns a random bit if $j'_* \neq j_*$. The winning probability of the adversary is the same as in G_2.

Game G'_3: This game is similar to G'_2, but from the moment when $n > j'_*$, each time the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, n)$ is called on a query (pk_n, m_n) and runs $c_n \leftarrow \mathsf{Enc}(\mathsf{pk}_n, \mathsf{k}_*, m_n, n; r_n)$, it replaces the following instructions $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, [\widehat{\mathsf{ek}}_{n-1}\|\widehat{m}_n\|\widehat{\mathsf{mk}}_n\|\widehat{v}_n\|\mathsf{pk}_n\|n]; \widehat{u}_n)$; by the sequence of instructions $\mathsf{str}_n \leftarrow \widehat{\mathsf{ek}}_n$, $[\widehat{\mathsf{ek}}_{n-1}\|\widehat{m}_n\|\widehat{\mathsf{mk}}_n\|\widehat{v}_n\|\mathsf{pk}_n\|n]$; $\mathsf{rnd}_n \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_n|}$; $\widehat{C}_n \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_n, \mathsf{rnd}_n; \widehat{u}_n)$;
Moreover, when it computes the challenge $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,0)}, l_*; r_*)$, the challenger replaces the instruction $\widehat{C}_{l_*} \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_{l_*}, [\widehat{\mathsf{ek}}_{l_*-1}\|\widehat{m}_{(*,b)}\|\widehat{\mathsf{mk}}_{l_*}\|\widehat{v}_{l_*}\|\mathsf{pk}_*\|l_*]; \widehat{u}_{l_*})$; by the sequence of instructions $\mathsf{str}_{l_*} \leftarrow [\widehat{\mathsf{ek}}_{l_*-1}\|\widehat{m}_{(*,b)}\|\widehat{\mathsf{mk}}_{l_*}\|\widehat{v}_{l_*}\|\mathsf{pk}_*\|l_*]$; $\mathsf{rnd}_{l_*} \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_{l_*}|}$; $\widehat{C}_{l_*} \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_{l_*}, \mathsf{rnd}_{l_*}; \widehat{u}_{l_*})$. We claim that:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G'_2] - \mathsf{Pr}[\mathcal{A} \text{ wins } G'_3]| \leq 2 \cdot (q_n + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

We proof this claim by using an hybrid argument. We define an hybrid game $G'_{3,i}$ as follows:

Game $G'_{3,i}$: If $i = q_n + 2$, then $G'_{3,i} = G'_2$, else if $1 \leq i \leq (q_n + 1)$, then the game $G'_{3,i}$ is the same as $G'_{3,i+1}$, but if $i > j'_*$ and $l_* \neq i$, then when the oracle $\mathsf{Enc}(\cdot, \mathsf{k}_*, \cdot, i)$ is called on a query (pk_i, m_i) and runs $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{k}_*, m_i, i; r_i)$, the challenger replaces the following instructions :
$\widehat{C}_i \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_i, [\widehat{\mathsf{ek}}_{i-1}\|\widehat{m}_i\|\widehat{\mathsf{mk}}_i\|\widehat{v}_i\|\mathsf{pk}_i\|i]; \widehat{u}_i)$;
by:
$\mathsf{str}_i \leftarrow [\widehat{\mathsf{ek}}_{i-1}\|\widehat{m}_i\|\widehat{\mathsf{mk}}_i\|\widehat{v}_i\|\mathsf{pk}_i\|i]$; $\mathsf{rnd}_i \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_i|}$; $\widehat{C}_i \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_i, \mathsf{rnd}_i; \widehat{u}_i)$;
Moreover, if $l_* = i$, then when it computes the challenge $c_{l_*} \leftarrow \mathsf{Enc}(\mathsf{pk}_*, \mathsf{k}_*, m_{(*,0)}, l_*; r_*)$, the challenger replaces the instruction:
$\widehat{C}_{l_*} \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_{l_*}, [\widehat{\mathsf{ek}}_{l_*-1}\|\widehat{m}_{(*,b)}\|\widehat{\mathsf{mk}}_{l_*}\|\widehat{v}_{l_*}\|\mathsf{pk}_*\|l_*]; \widehat{u}_{l_*})$;

by:
$\mathsf{str}_{l_*} \leftarrow [\widehat{\mathsf{ek}}_{l_*-1} \| \widehat{m}_{(*,b)} \| \widehat{\mathsf{mk}}_{l_*} \| \widehat{v}_{l_*} \| \mathsf{pk}_* \| l_*]$; $\mathsf{rnd}_{l_*} \xleftarrow{\$} \{0,1\}^{|\mathsf{str}_{l_*}|}$; $\widehat{C}_{l_*} \leftarrow \mathsf{SEnc}(\widehat{\mathsf{ek}}_{l_*}, \mathsf{rnd}_{l_*}; \widehat{u}_{l_*})$;
We claim that, for all $i \in [\![1, q_n + 1]\!]$:

$$\left|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G'_{3,i+1}\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G'_{3,i}\right]\right| \leq 2 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{SKE}}(\lambda).$$

This claim can be proven by reduction. We omit the details of this proof because it is very similar to the proof in the game $G_{3,i}$.

Game G'_4: This is the same game as G'_3, but $\widetilde{m}_{(*,b)}$, $\widetilde{\mathsf{mk}}_{l_*}$ and $\widetilde{v}_{l_*}$ are substituted by random bit-strings of same length. The winning probability of the adversary is the same as in G_4.

Finally, we have that $\mathbf{Adv}^{\mathsf{IND\text{-}CCA}_0}_{\mathsf{CHAPO}}(\lambda) = \mathbf{Adv}^{\mathsf{IND\text{-}CCA}_1}_{\mathsf{CHAPO}}(\lambda)$. This concludes the proof. □