Reducing the Cost of Authenticity with Leakages: a CIML2-Secure æ Scheme with One Call to a Strongly Protected Tweakable Block Cipher

Francesco Berti, Olivier Pereira, François-Xavier Standaert

ICTEAM/ELEN/Crypto Group
Université catholique de Louvain
B-1348 Louvain-la-Neuve, Belgium
emails: {francesco.berti,olivier.pereira,fstandae}@uclouvain.be

Abstract. This paper presents CONCRETE ($Commit-Encrypt-Send-the-Key$) a new Authenticated Encryption mode that offers CIML2 security, that is, ciphertext integrity in the presence of nonce misuse and side-channel leakages in both encryption and decryption.
CONCRETE improves on a recent line of works aiming at *leveled implementations*, which mix a strongly protected and energy demanding implementation of a single component, and other weakly protected and much cheaper components. Here, these components all implement a tweakable block cipher TBC.
CONCRETE requires the use of the strongly protected TBC only once while supporting the leakage of the full state of the weakly protected components – it achieves CIML2 security in the so-called *unbounded leakage model*.
All previous works need to use the strongly protected implementation at least twice. As a result, for short messages whose encryption and decryption energy costs are dominated by the strongly protected component, we halve the cost of a leakage-resilient implementation. CONCRETE additionally provides security when unverified plaintexts are released, and confidentiality in the presence of simulatable leakages in encryption and decryption.

Keywords: Leakage-resilience, authenticated encryption, leveled implementation, Ciphertext Integrity with Misuse and Leakage (CIML2).

1 Introduction

Authenticated encryption (AE) provides in a single scheme both confidentiality and authenticity. Nowadays AE is a standard primitive[16,30,32,34,9] (e.g., it is only the one accepted in TLS 1.3 [20]).
Although these schemes are secure in the black box model (that is, when adversaries have access only to the inputs and outputs), they may not be secure when implemented, because their implementation may leak via side-channels (e.g., the

computation time, the power consumption, or the electromagnetic radiation) some information about the secrets (as the key or the plaintext) involved in the cryptographic algorithm [1,24,27,28,26]. These attacks may affect the Internet-of-Things (IoT) since sensors, being deployed in a not-secure environment, are an easy target for side-channel attacks. Against such threats there are three types of countermeasures: hardware, software and protocol ones.
Hardware and software countermeasures (for example, noise addition, masking or hiding) aim to reduce the information leaked by the implementation, but they are very expensive and depend on technological assumptions, which may be false [36,11,27,29]. In particular, a good masking scheme has a great overhead, compared to a non protect implementation, which may be tens or hundred of times [18].
A complementary approach is to design schemes which are inherently more secure agaisnt side-channel attacks (for example, manipulating the plaintext and the key as little as possible, using a key only a few number of times before changing it) [17]. This protocol approach, called *leakage resilience*, try to minimize the use of the software and hardware countermeasures, without damaging the security.
In particular, since cryptographich algorithms are based on cryptographic primitives, like PRPs, PRFs and hash functions, a leakage resilient scheme may use them protected in the same way (*uniform implementation*) or it may assume that there is a weak protected implementation for all of them and, for only one, a very well protected implementation, called *leak free* for simplicity, which is very slow and, therefore, which should be used as little as possible. This is called a *leveled implementation* [33,11]. Although, actually, no implementation is leak free, we support this concept because, first, it models well protected components (e.g. a PRP with an high order masking), second, it shows where the efforts to protect against side-channel attacks should be concentrated.
Although leakage affects the two goals of *leakage resilient authenticated encryption* (LRAE), in this paper we care about authenticity only (for privacy we reuse previous works [33,11]). Our goal is to achieve CIML2 (*Ciphertext integrity with coin misuse and leakage on encryption and decryption*) in the unbounded leakage model.
CIML2, introduced by Berti et al. [13], assumes that the adversary receives the leakage of every encryption and decryption query he does. Moreover, the adversary has taken control of the random source used by the AE scheme. In the *unbounded leakage model* everything computed by the scheme is leaked apart from the key used in the leak free component (that is, all inputs and outputs of every component and all the keys used by non leak free components). In particular, this implies that to have CIML2 in this model, the tag *cannot* be recomputed during decryption, because, otherwise, it would be leaked.
By now, all modes [13,19,10] achieving CIML2 use at least 2 calls to the leak free component per encryption or decryption query. That is, if c_{LF} is the cost of a call to a leak free component, which is a block cipher, c_{wp} is the cost of a call to a weak protected component, which is another block cipher with the same block

size, the best cost to process a message of l blocks is $2c_{\mathsf{LF}} + 4lc_{wp}$ [10]. Thus, for short messages, which are usually those sent by sensors in the IoT, this cost is dominated by the cost of the leak free component. It is desiderable to reduce the number of calls to the leak free to one.
Moreover, constrained device may have to release unverified plaintexts (RUP), (that is, the decryption of a ciphertext before assessing its authenticity), due to little secure memory. This may be problematic [2,5,3].

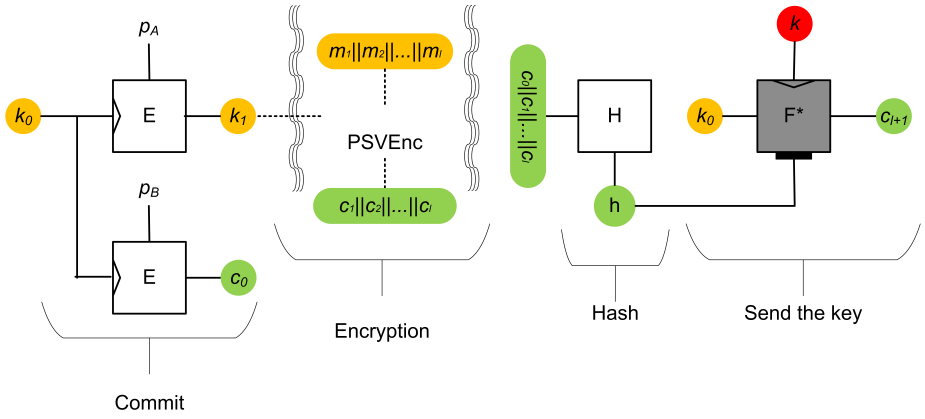

Fig. 1. The scheme CONCRETE *Commit − Encrypt − Send − the − Key*. We use red for long term secret, orange for ephemeral one and green for outputs. The gray shadowed component is the leak free. The key k_0 is randomly picked. It uses the PSV encryption scheme [33], described in Fig. 2. For decryption, first k_0 is recomputed $k_0 = \mathsf{F}_k^{*,-1}(h, c_{l-1})$, then, c_0 is recomputed and cheked. From k_0 decryption proceeds in the natural way.

Our contribution We provide a mode which is CIML2 secure and uses only one leak free call per execution in both encryption and decryption: CONCRETE(COmmit-eNCRypt-sEnd-The-kEy), see Fig. 1. The cost for l block message is $c_{\mathsf{LF}} + 4(l + 1)c_{wp}$. Thus, we, approximately, halve the cost when the message is short and the leak free is much more expensive than the weak protected component.
The key idea is to use a random string k_0 which is kept secret (as in [3,6,11]) as the key used to encrypt the message, obtaining $c_1, ..., c_l$. Moreover, the ciphertext contains a commitment of k_0, c_0, and an encryption of k_0, c_{l+1} which depends on all other values $(c_0, ..., c_l)$. As the previous modes achieving CIML2 (for example, [13]), the leak free component is a strong tweakable pseudorandom permutation, which is inverted during the decryption.
We also remove the hypothesis of range-oriented pre-image resistance for the hash function to obtain CIML2.
In addition our scheme is an AE secure scheme which is secure when unverified plaintexts are released (RUPAE) and provides privacy in the presence of leakage (CPAL and CCAL [19]).

Other works The security definitions in the presence of leakage are given in the works of Guo et al. [19] and Barwell et al. [4]. While Barwell et al. allow all components to leak, but not too much, that is, they assume an uniform implementation. Guo et al. give the definitions in the leveled setting. Moreover, they use different leakage models (the unbounded for authenticity and the simulatable one for confidentiality, introduced in [37]). Since we assume a leveled implementation, we use the definitional framework of Guo et al. [19].
With respect to the leakage resilient construction in a leveled implementation, Pereira et al. [33] provide a leakage-resilient encryption mode (PSV), which uses only one leak free. Berti et al. [11] used this mode as a key ingredient to provide a leakage resilient AE mode which is misuse-resistant (called DTE) and CIML secure (that is, CIML2 when decryption does not leak), which uses two leak free calls. It has been modified by Berti et al. [13] in DTE2 to provide CIML2. Moreover, they provide EDT, another CIML2-secure mode (both of them use two leak free calls). Guo et al. [19] provide a mode which is CIML2-secure and misuse resistant (it uses still two leak free calls, but it uses more calls to the weak protected components). Berti et al. [10] propose TEDT a scheme which is CIML2 secure, in addition it is beyond birthday secure and provides multi-user security (it, still, uses two call to the leak free components).
Dobraunig et al. [15] and Bertoni et al. [14] proposed two LRAE designs based on sponge. Although their solutions are very interesting, there is an approximate comprehension about leakages of sponges. Moreover, CIML2 in the unbounded model seems unachievable with a construction based only on sponges, since it seems impossible not to recompute the tag during decryption (differently from what it is done in all CIML2 secure modes).
Finally, Barwell et al. [4] proposed an LRAE mode based on a uniform implementation. Moreover, for confidentiality, they assume that the challenge query of the CPA and CCA game does not leak.

Structure of the paper We review in Sec. 2, the main definitions and notations used in the paper. Then, in Sec. 3 we presented the specifications for our scheme and the structure of previous constructions. After that, we present the rationale of the design of CONCRETE. The security proofs of CIML2, AE, RUPAE, CPAL2 and CCAL2 conclude the paper. For space constraint, there is only a sketch of the proofs, they can be found with all the details in the appendix, with an extended background, a detailed analysis of previous works and the extension to associated data.

2 Background

Notations We use $(q_1, ..., q_d, t)$- bounded adversaries who have access to the oracles $\mathcal{O}_1, ..., \mathcal{O}_d$ and can make at most q_i queries to oracle $\mathcal{O}_i$ and who runs in time bounded by t. If $\mathcal{O}$ is an oracle, $\mathcal{O}\mathsf{L}$ is its leaking variant.
Given a string x, we denote with $|x|$ its length. With $\{0,1\}^n$ we denote the set of all n bits long stings, with $\{0,1\}^{\leq n}$ the set of all at most n bits long strings

(that is, $\{0,1\}^{\leq n} = \bigcup_{i=1}^{n} \{0,1\}^i$). 0^j denotes the string composed by n zeros.

We denote with $x \xleftarrow{\$} \mathcal{X}$ the fact that the element x is picked uniformly at random from the set $\mathcal{X}$.

2.1 Primitives: Hash functions, PRFs and STPRPs

A *hash function* is a mapping $\mathsf{H} : \mathcal{S} \times \mathcal{M} \longmapsto \mathcal{B}$. We suppose that hash functions are $(0,t)$-*collision resistant*, that is, for any $(0,t)$ adversary the probability that he, given the key s of the hash function, outputs a collision, (that is, two different $m_0, m_1 \in \mathcal{M}$ s.t. $\mathsf{H}_s(m_0) = \mathsf{H}_s(m_1)$) is bounded by ϵ. From now, since the adversary knows the key of the hash functions, we omit the key s. Thus, for simplicity we refer to the hash function H.
A (q,t,ϵ)-*pseudorandom function* (PRF) is a mapping $\mathsf{E} : \mathcal{K} \times \mathcal{M} \longmapsto \mathcal{T}$ s.t. there is no (q,t)-adversary able to distinguish with probability better than ϵ, if he is interacting with $\mathsf{E}_k 9\cdot)$ for a random key or with a random function $\mathsf{f}(\cdot)$ with the same signature as $\mathsf{E}_k(\cdot)$.
A (q,t,ϵ)- *strong tweakable pseudorandom function* (STPRP) is a mapping $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{M} \longmapsto \mathcal{T}$ s.t. $\forall (k,tw) \in \mathcal{K} \times \mathcal{TW}$ $\mathsf{F}_k^{tw}(\cdot)$ is a permutation and there is no (q,t)-adversary able to distinguish with probability better than ϵ, if he is interacting with $\mathsf{F}_k(\cdot,\cdot)$ and $\mathsf{F}_k^{-1}(\cdot,\cdot)$ for a random key or with a random permutation f with the same signature as $\mathsf{F}_k(\cdot,\cdot)$[25]. More details in App. A.2.

2.2 Authenticated encryption (AE)

For authenticated encryption (AE) schemes we use the syntax of encryption schemes introduced in Katz and Lindell [22], which was used also in many works about AE [7,23]. The reason we do not use the nAE notion (see Appendix(Def.10) will be clear when scheme CONCRETE is presented.

Definition 1. *An* authenticated encryption scheme AE *is a triple of algorithm* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a probabilistic algorithm which takes as input the tuple* $(k,m) \in \mathcal{K} \times \mathcal{M}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k(m)$. *The decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k,c) \in \mathcal{K} \times \mathcal{C}$ *and outputs a string* $m \leftarrow \mathsf{Dec}_k(c)$ *which is either a string in* $\mathcal{M}$ *or the symbol "*$\perp$*" (invalid). We require that the algorithm* Enc *and* Dec *are the inverse of each other, asking the correctness and tidiness property. If* $m \leftarrow \mathsf{Dec}_k(c)$ *with* $m \neq \perp$ *we say that* c *is* valid*, otherwise it is* invalid*.*

2.3 Security for Authenticated Encryption

The security definition we require to AE schemes is:

Definition 2. *An authenticated encryption* AE *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-AE *secure if the following advantage*

$$\mathsf{Adv}^{\mathsf{AE}}_{\Pi,\mathsf{A}} := \left| \Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right] \right| \leq \epsilon$$

for any (q_E, q_D, t)-adversary A, where the key k is picked uniformly at random, the algorithm $\$(m)$ answers a random string of length $|c|$ with $c \leftarrow \mathsf{Enc}_k(m)$, and $\perp(\cdot)$ is an algorithm which answers always $\perp$ ("invalid"). The adversary A may ask q_E encryption queries (to the left oracle) and q_D decryption queries (to the right oracle). If he receives c as an answer of the first oracle, [that is, $c \leftarrow \mathsf{Enc}_k(m)$ (or $c \leftarrow \$(m)$)] he is not allowed to query the second oracle on input c.

It provides both confidentiality and authenticity. It means, also, that Enc internally has a way to produce some randomness. More details in App. D.2.

2.4 Ciphertext integrity with misuse and leakage in encryption and decryption

Berti et al. [13] introduced ciphertext integrity with misuse and leakage in encryption and decryption. Again, we adapt it to the syntax used in Def. 1.

Definition 3. *An authenticated encryption (AE) scheme $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ is (q_E, q_D, t, ϵ)-*ciphertext integrity with misuse and leakage in encryption and decryption *($\mathsf{CIML2}$)-secure if, for all (q_E, q_D, t)-bounded adversaries, we have*

$$\Pr\left[\mathsf{A}^{\mathsf{EncL}(\cdot),\mathsf{DecL}(\cdot)} \Rightarrow c^* \text{ s.t. } c^* \text{ is fresh and valid} \mid \right] \leq \epsilon$$

[c^* fresh means that it is not obtained an an answer of an EncL query]

According to our AE syntax (see Def. 1), we have to define misuse. We suppose that the adversary has taken control of the random source (which may be, for example, a pseudorandom generator). Thus, for us, misuse means that the adversary chooses and provides the randomness to the encryption oracle, which, thus, is now deterministic.

Without leakage, it is irrelevant if the adversary has or not has access to the decryption oracle [22,12]. This is not the case in the presence of leakage [11].

Leakage model: the unbounded model We suppose that every input, output and key is leaked apart the key of the leak free STPRP F^*. This is the *unbounded model*, which is quite liberal [11].

A more formal treatment of $\mathsf{CIML2}$ and of the unbounded model can be found in App. A.6.
Since the core of this contribution is authenticity with leakage, we leave the definitions of security in a RUP (Release of unverified plaintext) scenario (RUPAE) and confidentiality with leakage ($\mathsf{CPAL2}$ and $\mathsf{CCAL2}$) in the Appendix(App. A).

3 Design constraints and previous solutions

Notations For F^* and E, we assume $\mathcal{K} = \mathcal{M} = \mathcal{TW} = \mathcal{T} = \{0,1\}^n$. The message space for Enc is $\{0,1\}^{\leq Ln}$.
We call a *block* a n bit string. Given a message m, we parse it in $(m_1, ..., m_l)$ with $|m_1| = ... = |m_{l-1}| = n$ and $|m_l| \leq n$ (By abuse of notation m_l is also called a block).

The design constraints of the scheme are the following:
- AE secure in the black box model
- $\mathsf{CIML2}$ secure in the unbounded model
- CPAL and CCAL secure with some hypothesis about the leakage
- only one leak free call per execution
- the key k of the scheme used only as a key of the leak free
- E changes keys as much as possible (*rekeying*)

The hypothesis on k is due to the fact that the key of the leak free is the only internal secret not leaked in the unbounded model.
To reduce the possibility of leakage, the PRF E should not be used with more than two different plaintexts for any key it uses in any security game.
There is such an encryption scheme, called PSV (see Fig. 2) [33] and some AE scheme [11,13,19,10], based on the PRG proposed by Standaert et al. [37], which use a PRF E and which is based on *rekeying*. At every round, this PRG changes the key, called *ephemeral*, the PRF E is used with, and it is very hard that the same key is used in two different rounds.
The problem how to use PSV in a bigger scheme is how to provide the first ephemeral key. Usually, LRAE schemes based on PSV may be divided in three parts, not necessarily in this order:
1 Generation of the first ephemeral key (for us k_1)
2 Encryption, using PSV starting from k_1
3 Authentication
The leak free calls are usually used in Step 1 and 3, and the verification should be done in a clever way, not simply a basic recheck of the authentication part.

A more detailed analysis of one of this schemes EDT [13] with a formal treatment of a rekeying scheme, can be found in App. C.

4 Design rationale of the Commit-Encrypt-Send-the-Key CONCRETE transformation

We present here the ideas of $\mathsf{CONCRETE}$(Fig. 1). In particular, how we treated each of the three different parts of a LRAE scheme:
- **Derivation of the first ephemeral key** We pick a randomly k_0 as first ephemeral key. We use a round the PRG of Standaert et al. [37] to obtain a commit of k_0 (called c_0) and a new fresh key (k_1). That is, using the

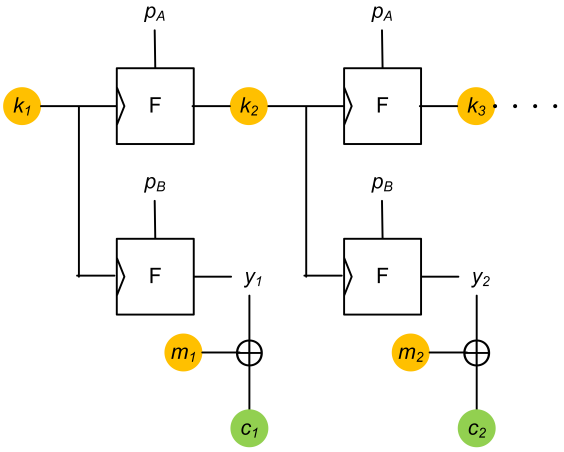

Fig. 2. The PSV encryption algorithm, presented at CCS 2015 by Pereira et al. [33].

public constants p_A and p_B (two strings of n bits, with $p_A \neq p_B$) we obtain $k_1 = \mathsf{E}_{k_0}(p_A)$ and $c_0 = \mathsf{E}_{k_0}(p_B)$.

- **Encryption** From k_1 the PSV [33] (see Fig. 2) encryption algorithm is used to encrypt m, using the constants p_A and p_B, obtaining $c_1, ..., c_l$. We denote the algorithm in this part enc.
- **Authentication** Since k_0 is picked uniformly at random, it must be recomputed by the decryption algorithm Dec from the ciphertext c. Thus, to send it, we hash the commitment c_0 and the output of the encryption part $c_1, ..., c_l$ obtaining $h = \mathsf{H}(c_0 \| c_1 \| ... \| c_l)$ and we encrypt k_0 obtaining $c_{l+1} = \mathsf{F}^*_k(h, k_0)$. The ciphertext is $c := (c_0, c_1, ..., c_l, c_{l+1})$.
- **Decryption** First k_0 is retrieved, with $k_0 = \mathsf{F}^{*,-1}_k(h, c_{l+1})$ with $h = \mathsf{H}(c_0 \| ... \| c_l)$, then $\tilde{c_0} = \mathsf{E}_{k_0}(p_B)$ is computed. If $c_0 = \tilde{c_0}$, the ciphertext is deemed valid and decryption proceeds in the natural way; otherwise, the ciphertext is deemed invalid.
- **Ciphertext expansion** The ciphertext has an exapnsion of two blocks, that is, given $c \leftarrow \mathsf{Enc}_k(m)$, $|c| = |m| + 2n$.
- **Cost** If the hash function is implemented with the Hirose construction [10], thus its cost is $2c_E$ for each block processed, then, the cost of CONCRETE to process l block message is $c_{\mathsf{F}^*} + 4(l+1)c_{\mathsf{E}}$, with c_{F^*} and c_{E} the cost, respectively, of one call to F^* and E.

A detailed description of the scheme can be found in Tab. 4 in the Appendix. Note that $c_0, ..., c_l$ can be seen as $\mathsf{PSV}_{k_0}(0^n \| m)$.

Cautionary note It is possible, as discussed in the proofs, to replace PSV with other encryption scheme based on rekeying. The security constraints of such replacements are discussed after the proof in the Appendix.
Although CONCRETE is AE and CIML2 secure and it uses only one leak free call, it is neither nonce-misuse resistance (as DTE [11] and FEMALE [19]) or secure beyond birthday and in the multi-user case (as TEDT [10]).

5 Security results for CONCRETE

For clarity, in this section we do not consider the time bounds. They will be discussed in the Appendix, where the theorems are restated.

Notations We introduce some definitions and notations to make the proofs lighter. We use superscripts to indicate to which encryption and decryption query the message (or the ciphertext) is referred to.
Given a ciphertext $c = (c_0, ..., c_l, c_{l+1})$ we define the *partial ciphertext* as the vector $(c_0, ..., c_l)$, that is, the ciphertext without considering the block c_{l+1} encrypting the key.
A value is *fresh* if it has never appeared before in the history of the game.

5.1 CIML2 security

Before proving the CIML2 security for CONCRETE, we have to define the leakage function in the unbounded model. We observe that $\mathsf{L}_E(r, m; k) := k_0 = r$, that is the leakage is the randomness r used by $\mathsf{Enc}_k(\cdot)$, because, from it all values, not given in the ciphertext, can be recomputed apart from k.
On the other hand, $\mathsf{L}_D(c; k) := k_0$, because from k_0 the adversary is able to recompute all values used in the decryption apart from k. Interestingly, when there is misuse, that is, the adversary has taken control of the random generator providing the randomness r, there is no leakage during encryption queries, since r is already known to the adversary.

Theorem 1. *Let* F^* *be a* $(Q+1, \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudorandom permutation (*STPRP*), let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* $(0, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Then, the mode* CONCRETE*, which encrypts messages which are at most L-block long, is* (q_E, q_D, ϵ)*-*CIML2 *secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{(q_D+1)(L+1)(q_D+2q_E)}{2^{n+1}} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}}.$$

Observation on the bound We want to discuss some terms of the bound:
- ϵ_{CR} because, if there is a collision, the mode is broken,
- $(q_D+1)\epsilon_{\mathsf{PRF}}$, because we do not check k_0, but $E_{k_0}(p_B)$. Otherwise, the mode would not be AE secure.
- $\frac{(q_D+1)(L+1)(q_D+2q_E)}{2^{n+1}}$ because we need that, in every decryption query, k_0 must have not been used as ephemeral key. It may be improved to $\frac{(q_D+1)(q_D+2q_E)}{2^{n+1}}$ if E is not used in PSV, [for example, we may use a different PRF (this choice requires one more component to be implemented)].

Proof (Sketch). In the proof, first, we replace F^* with a random tweakable permutation, then, we suppose that all the hash outputs are different (provided that their inputs are different). For fresh decryption queries, on input $c = (c_0, c_1, ..., c_l, c_{l+1})$ we observe the following:

1. If the partial ciphertext $(c_0, ..., c_l)$ is fresh, then, its hash h is fresh. Thus, k_0 is random. Consequently, the probability that $c_0 = \mathsf{E}_{k_0}(p_B)$ is negligible.
2. If the partial ciphertext $(c_0, ..., c_l)$ is not fresh and comes from an encryption query, then, the couple $((c_0, ..., c_l), c_{l+1})$ must be fresh; otherwise, either the decryption query is not fresh [not possible by hypothesis] or it is the repetition of a previous decryption query [so, its validity has already been established]. Thus, $k_0 = \mathsf{F}_k^{*,-1}(h, c_{l+1})$ is still random. Then, again, the probability that $c_0 = \mathsf{E}_{k_0}(p_B)$ is negligible.

To prove that $\Pr[c_0 = \mathsf{E}_{k_0}(p_B)]$ is negligible, we use that E is a PRF, thus, we must assume that k_0 is fresh.

The complete proof can be found in App D.1 as well with the theorem with the time bounds (Thm. 6).

5.2 AE security

After having proved the authenticity, we want to prove the confidentiality, which should be based on the security of the PSV encryption scheme. We start studying confidentiality in the blackbox model:

Theorem 2. *Let* F^* *be a* $(Q, \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudorandom permutation (*STPRP*), let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* $(0, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Then, the mode* $\mathsf{CONCRETE}$*, which encrypt messages which are at most* L*-block long, is* (q_E, q_D, ϵ)*-*AE *secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_D(L+1)(q_D+2q_E)}{2^{n+1}} + \frac{q_D}{2^n}$$
$$(q_E(L+1)+q_D)\epsilon_{\mathsf{PRF}} + \frac{q_E(L+1)[q_E(L+1)-1]}{2^{n+1}} + \frac{(q_D+q_E)(q_D+q_E-1)}{2^{n+1}}.$$

Observation on the bound In addition to the bound due to the $\mathsf{CIML2}$ security (which is the same as for the ciphertext integrity) we have:
- $\epsilon_{\mathsf{STPRP}} + \frac{(q_E+q_D)(q_E+q_D-1)}{2^{n+1}}$ because F^* is a STPRP and not a PRF,
- $(q_E(L+1))\epsilon_{\mathsf{PRF}}$ is due to PSV,
- $\frac{q_E(L+1)[q_E(L+1)-1]}{2^{n+1}}$ because we need that, in every encryption query, all keys used by E are different.

Proof (Sketch). First, we observe that the scheme is ciphertext-integrity secure (since it is $\mathsf{CIML2}$ secure), then, we observe that all the ciphertext blocks can be replaced by random ones since either they are obtained via a STPRP with a different input (c_{l+1}) or via the PSV encryption scheme using a different key k_0 per encryption query.

The complete proof, the theorem with the time bounds (Thm. 7) and a discussion of what happens if PSV is replaced with another scheme can be found in App. D.2.

5.3 The RUPAE security

Even if unverified plaintexts are released, CONCRETE is secure:

Theorem 3. *Let* F^* *be a* $(Q, \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudorandom permutation (*STPRP*), let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* $(0, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Then, the mode* CONCRETE*, which encrypt messages which are at most* L*-block long, is* (q_E, q_D, ϵ)-RUPAE *secure with* ϵ *bounded by*

$$\epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + Q(L+1)\epsilon_{\mathsf{PRF}} + \frac{q_D}{2^n} + \frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}} + \frac{q_E(q_E-1)}{2^{n+1}}$$

with $Q = q_E + q_D$*.*

Observation on the bound In addition to bounds due to the previous theorems, we have

- $Q(L+1)\epsilon_{\mathsf{PRF}}$ due to PSV,
- $\frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}}$, because we suppose that every ephemeral key used in an encryption or decryption query are different,
- $\epsilon_{\mathsf{STPRP}} + \frac{q_E(q_E-1)}{2^{n+1}}$ because F^* is a STPRP and not a PRF (a part of the bound is in the previous term)

Proof (Sketch). We have already proved the CIML2 (thus, the ciphertext integrity) and the AE security. To prove the RUPAE is enough to observe that, for invalid ciphertexts, the k_0 obtained is random. Moreover, from a random k_0, PSV gives a random decryption.

The complete proof, the theorem with the time bounds (Thm. 8) and a discussion of what happens if PSV is replaced with another scheme can be found in App. D.3.

5.4 CPAL2 and CCAL2 security

In this section, we study confidentiality with leakage of CONCRETE. Similarly to what done for PSV [33], we reduce the whole security of the scheme to the EavL2s security of an ideal version of PSV^I, where the PRF $\mathsf{E}_{k_i^j}$ is replaced with a random function and the leakage of the $\mathsf{E}_{k_i^j}$ with simulated leakages, which encrypts only one block message (see Tab. 1).
The EavL2s game (see Tab. 1) is a game where the adversary chooses two different one block message and receives the encryption and the leakage of one of them, and he has to guess what message has been encrypted. (A scheme is $(q_{\mathsf{L}}, \epsilon)$-EavL2 secure if the probability he correctly guesses is bounded by $\frac{1}{2} + \epsilon$.

Theorem 4 (informal). *Let* F^* *be a* $(q_E + 1, \epsilon_{\mathsf{STPRP}})$-STPRP*, whose implementation is leak free, let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$-PRF*, whose implementation has some leakage property, let* $\mathsf{PSV}s^I$ *be* $(q_{\mathsf{L}}, \epsilon_{\mathsf{EavL2s}})$-EavL2*-secure, then* CONCRETE*, if it encrypts at most* L *block messages, is* (q_E, ϵ)-CPAL2*-secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \frac{q_E}{2^n} + \epsilon_{\mathsf{2\text{-}sim}'} + (L+1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{EavL2s}})$$

The EavL2s game and the idealized variant of PSV which encrypts a single block, PSVs^I

EavL2s	PSVs^I
Initialization: $k_0 \xleftarrow{\$} \mathcal{K},\ p_A, p_B \xleftarrow{\$} \{0,1\}^n$ $b \xleftarrow{\$} \{0,1\}$	Gen^I : $k_0 \xleftarrow{\$} \{0,1\}^n$
Challenge output: $(m^{*,0}, m^{*,1}) \leftarrow \mathsf{A}^{\mathsf{L}}(p_A, p_B)$ $(c, k_1, \mathsf{L}_{\mathsf{PSVs}^I}) \leftarrow \mathsf{encs}\mathsf{L}_{k_0}(m^{*,b})$	$\mathsf{encs}^I_{k_0}(m)$: $y \xleftarrow{\$} \{0,1\}^n$ $c = y \oplus m$ $k_1 \xleftarrow{\$} \{0,1\}^n$ Return (c, k_1)
Finalization: $b' \leftarrow \mathsf{A}^{\mathsf{L}}(c, k_1, \mathsf{L}_{\mathsf{PSVs}^I})$ If $b = b'$ Return 1, Else Return 0	$\mathsf{decs}^I_{k_0}(c)$ proceeds in the natural way.
The leakage resulting from $\mathsf{encs}^I(m)$ is defined as $\mathsf{L}_{\mathsf{encs}^I}(m, k_1, y; k_0) :=$ $(\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1), \mathcal{S}^{\mathsf{L}}(k_0, p_B, y), \mathsf{L}_{\oplus}(m, y),\ \mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-)$ with $k^- \xleftarrow{\$} \{0,1\}^n$.	

Table 1. The EavL2s experiment and the idealized single block version PSV^I. Note that k_1 is given as output for composability as used in App. D.4

About the bound we can observe:
- $\epsilon_{\text{2-sim}'} + L\epsilon_{\text{2-sim}}$ is due to the leakage assumptions,
- $L(\epsilon_{\text{2-sim}} + \epsilon_{\mathsf{EavL2s}})$ is the EavL2-security of PSV [33].

The leakages assumptions we use are presented in App. A.5, while the theorem with the time bounds (Thm. 9) and the proof in App. 9.

Proof (Sketch). First, we reduce the EavL2 security of CONCRETE to the EavL2s security of PSVs^I, using the same argument as Pereira et al. [33] (we have to do a little tweak in their proof to consider the additional leakage source of c_0 and c_{l+1}).
Then, we replace the STPRP F^* with a random function, and we replace its leakage $\mathsf{L}_{\mathsf{F}^*}(\cdot,\cdot;\cdot)$ with a simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(\cdot,\cdot,\cdot,\cdot)$; after that, observing that, since, k_0 is randomly picked, the leakage of other encryption queries do not give any more information about the challenge query, we reduce the CPAL2 adversary to an EavL2 adversary.

Moreover, CONCRETE is CCAL2 secure:

Theorem 5 (informal). *Let* F^* *be a* $(q_E + q_D + 1, \epsilon_{\mathsf{STPRP}})$-STPRP, *whose implementation is leak free, let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$-PRF, *whose implementation has some leakage property, let* H *be a* $(0, \epsilon_{\mathsf{CR}})$*-collision resistant hash function, let* $\mathsf{PSV}s^I$ *be* $(q_{\mathsf{L}}, \epsilon_{\mathsf{EavL2s}})$-EavL2*-secure, then* CONCRETE, *if encrypts at most* L *block messages, is* (q_E, q_D, ϵ)-CCAL2*-secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_E + q_D}{2^n} + \frac{q_D(L+1)(q_D + 2q_E)}{2^{n+1}} +$$

$$(q_D + L + 1)\epsilon_{\mathsf{PRF}} + \epsilon_{\text{2-sim}'} + L(\epsilon_{\text{2-sim}} + \epsilon_{\mathsf{EavL2s}})$$

With respect to the CPAL2 bound, we have added only the CIML2 bound.

Proof (Sketch). We reuse the proof of the CPAL2 security (Thm. 4). We add only that, due to the CIML2-security in the unbounded model, the adversary can only due invalid decryption queries and invalid decryption queries may not give any information about the challenge query, because the ephemeral k_0^* picked during the challenge query is independent from the ephemeral key k_0 recomputed during their decryptions.

The leakages assumptions we use are presented in App. A.5, while the theorem with the time bounds (Thm. 10) and the complete proof in App. 10.

6 Conclusion

With CONCRETE we have provided the first AE scheme achieving CIML2 in the unbounded model with only one leak free call. It provides also RUPAE and CPAL2 and CCAL2. This is a major improvement.
Its security is based crucially on the leak free assumption. Thus, try to keep the CIML2 security using a more liberal assumption on the leak free, like unpredictability with leakage, is an interesting scope for further research.

References

1. Martin R. Albrecht and Kenneth G. Paterson. Lucky microseconds: A timing attack on amazon's s2n implementation of TLS. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 622–643. Springer, 2016.
2. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.
3. Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.
4. Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

5. Guy Barwell, Daniel Page, and Martijn Stam. Rogue decryption failures: Reconciling AE robustness notions. In Jens Groth, editor, *Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings*, volume 9496 of *Lecture Notes in Computer Science*, pages 94–111. Springer, 2015.
6. Mihir Bellare. Symmetric ecryption revised. Technical report, 2018. https://spotniq.files.wordpress.com/2018/07/spotniq18-se-revisited.pdf.
7. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.
8. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Vaudenay [38], pages 409–426.
9. Dan J Bernstein. Caesar call for submissions, final, january 27 2014.
10. Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resilient aead mode for high (physical) security applications. 2019. https://eprint.iacr.org/2019/137.
11. Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 37–50. ACM, 2018.
12. Francesco Berti, Olivier Pereira, and Thomas Peters. Reconsidering generic composition: The tag-then-encrypt case. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings*, volume 11356 of *Lecture Notes in Computer Science*, pages 70–90. Springer, 2018.
13. Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.
14. Guido Bertoni, Joan Daemen, Michaël Peters, Gilles Van Assche, and Ronny Van Keer. Caesar submission: Ketje v2. Technical report, 2016. https://keccak.team/files/Ketjev2-doc2.0.pdf.
15. Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - towards side-channel secure authenticated encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.
16. Morris J Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. Technical report, 2007.
17. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 293–302. IEEE Computer Society, 2008.
18. Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.

19. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage-resilient authenticated encryption with misuse in the leveled leakage setting: Definitions, separation results, and constructions. *IACR Cryptology ePrint Archive*, 2018:484, 2018.
20. IETF. The transport layer security (tls) protocol version 1.3 draft-ietf-tls-tls13-28. Technical report, 2018. https://tools.ietf.org/html/draft-ietf-tls-tls13-28.
21. Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643. Springer, 2017.
22. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.
23. Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer, 2000.
24. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
25. Moses Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.
26. Jake Longo, Elke De Mulder, Dan Page, and Michael Tunstall. Soc it to EM: electromagnetic side-channel attacks on a complex system-on-chip. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 620–640. Springer, 2015.
27. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
28. Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.
29. Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
30. David A. McGrew. An interface and algorithms for authenticated encryption. *RFC*, 5116:1–22, 2008.
31. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances*

in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2014.
32. Yoav Nir and Adam Langley. Chacha20 and poly1305 for IETF protocols. *RFC*, 7539:1–45, 2015.
33. Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 96–108. ACM, 2015.
34. Thomas Peyrin and Yannick Seurin. Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 33–63. Springer, 2016.
35. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Vaudenay [38], pages 373–390.
36. Adi Shamir. Protecting smart cards from passive power analysis with detached power supplies. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 71–77. Springer, 2000.
37. François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.
38. Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.
39. Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 141–151. ACM, 2010.

A Extended background

A.1 Extended notation

Here we present the additional notation used in the proofs.

Given the sets $\mathcal{A}$ and $\mathcal{B}$,let $\mathcal{FUNC}(\mathcal{A},\mathcal{B})$ be the set of all functions $f : \mathcal{A} \longmapsto \mathcal{B}$.
With $\Pr[\mathsf{A}(x)^{\mathcal{O}_1(\cdot),...,\mathcal{O}_d(\cdot)} \Rightarrow y|Z]$ we denote the probability that the adversary with input x outputs y provided that event Z happens. This adversary has access to the oracles $(\mathcal{O}_1(\cdot), ..., \mathcal{O}_d(\cdot))$.

Given a string x of length l we denote with $\pi_{l'}(x)$ (with $l \leq l'$), the string composed by the first l, bits of x, that is, $\pi_{l'}(x)$ is the string x with the last $l - l'$ bits dropped.

A.2 Primitives: Hash functions, PRFs and STPRPs

First, we define collision resistant hash functions:

Definition 4. *A $(0, t, \epsilon)$-collision resistant hash function $\mathsf{H} : \mathcal{S} \times \mathcal{M} \longmapsto \mathcal{B}$ is a function such that, for every $(0, t)$-bounded adversary A, the probability that $\mathsf{A}(s)$ outputs a couple of distinct messages $(m_0, m_1) \in \mathcal{M}^2$, $m_0 \neq m_1$, such that $\mathsf{H}_s(m_0) = \mathsf{H}_s(m_1)$ is bounded by ϵ when $s \xleftarrow{\$} \mathcal{S}$ is picked uniformly at random, that is:*

$$\Pr[\mathsf{A}(s) \Rightarrow (m_0, m_1) \ s.t. \ m_0 \neq m_1, \mathsf{H}_s(m_0) = \mathsf{H}_s(m_1) | s \xleftarrow{\$} \mathcal{S}] \leq \epsilon$$

We call such a couple (m_0, m_1) a *collision* for H_s.
From now, since the adversary knows the key of the hash functions, we omit the key s. Thus, for simplicity we refer to the hash function H.

For completeness, since we talk about scheme which are CIML2 secure, we introduce the notion of *range oriented pre-image resistant hash function*:

Definition 5. *A $(1, t, \epsilon)$-range-oriented preimage resistant hash function $\mathsf{H} : \mathcal{S} \times \mathcal{M} \longmapsto \mathcal{B}$ is a function such that, for every (t)-bounded adversary A, the probability that $\mathsf{A}(s, y)$, for a y picked uniformly at random in $\mathcal{B}$, outputs a message $(m) \in \mathcal{M}$, such that $\mathsf{H}_s(m) = y$ is bounded by ϵ when $s \xleftarrow{\$} \mathcal{S}$ is picked uniformly at random, that is:*

$$\Pr[\mathsf{A}(s, y) \Rightarrow (m) \ s.t. \ \mathsf{H}_s(m) = y \ | s \xleftarrow{\$} \mathcal{S}, \ y \xleftarrow{\$} \mathcal{B}] \leq \epsilon$$

Berti et al. [13] presented a definition where the adversary has access to multiple targets (these two definitions are polynomially equivalent).

Then, we define pseudorandom functions:

Definition 6. *A function $\mathsf{E} : \mathcal{K} \times \mathcal{M} \longmapsto \mathcal{T}$ is a (q, t, ϵ)-*pseudorandom function *(PRF) if for any (q, t)-bounded adversary, the advantage:*

$$\mathsf{Adv}_{\mathsf{A}}^{\mathsf{PRF}} := \left| \Pr\left[\mathsf{A}^{\mathsf{E}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\mathsf{f}(\cdot)} \Rightarrow 1\right] \right| \leq \epsilon$$

with k and f picked uniformly at random from their domains, respectively $\mathcal{K}$ and the set of functions $\mathcal{FUNCT}(\mathcal{M}, \mathcal{T})$.

If for every k, $\mathsf{E}_k(\cdot)$ is a permutation and f is picked from set of permutations, the function E is called a pseudorandom permutation PRP.

To add more diversity and flexibility in the use of PRFs, Liskov et al. [25] introduce tweakable pseudorandom permutations. We need a strong ones, that is ones which are secure even if the adversary has access to their inverse:

Definition 7. *A function* $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{M} \longmapsto \mathcal{M}$ *is a* (q, t, ϵ)- strong tweakable pseudorandom function *(*STPRP*) if for any* (q, t)*-bounded adversary, the advantage:*

$$\mathsf{Adv}^{\mathsf{STPRP}}_{\mathsf{A}} := \left| \Pr\left[\mathsf{A}^{\mathsf{F}_k(\cdot,\cdot),\mathsf{F}_k^{-1}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\mathsf{f}(\cdot,\cdot),\mathsf{f}^{-1}(\cdot,\cdot)} \Rightarrow 1\right]\right| \leq \epsilon$$

with k *and* f *picked uniformly at random from their domains, respectively* $\mathcal{K}$ *and the subset of the set* $\mathcal{FUNC}(\mathcal{TW} \times \mathcal{M}, \mathcal{M})$ *composed by the functions* $\mathsf{f}(\cdot,\cdot)$ *s.t. for every tweak* $tw \in \mathcal{TW}$*,* $\mathsf{f}(tw, \cdot) : \mathcal{M} \longmapsto \mathcal{M}$ *is a permutation.*

For tweakable strong pseudorandom permutations (STPRPs), we use the following notation: we suppose that the adversary A has access to only one oracle which is either implemented with $(\mathsf{F}_k(\cdot,\cdot), \mathsf{F}_k^{-1}(\cdot,\cdot))$ for a random key k, or with $(\mathsf{f}(\cdot,\cdot), \mathsf{f}^{-1}(\cdot,\cdot))$. Thus, when he queries his oracle, A uses an additional input which is ± 1. When he queries on input $(tw, x, 1)$ it means A queries the left oracle (either $\mathsf{F}_k(\cdot,\cdot)$ or $\mathsf{f}(\cdot,\cdot)$) on input (tw, x), when on input $(tw, x, -1)$ it means A queries the right oracle (either $\mathsf{F}_k^{-1}(\cdot,\cdot)$ or $\mathsf{f}^{-1}(\cdot,\cdot)$) on input (tw, x).

We call hash functions, PRFs and STPRPs *primitives.*

A.3 Authenticated Encryption

We presented the definition of Authenticated Encryption in Def. 1, which uses a probabilistic encryption algorithm. We restate it for clarity:

Definition 8. *An* authenticated encryption scheme *(*AE*) is a triple of algorithm* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a probabilistic algorithm which takes as input the tuple key, message* $(k, m) \in \mathcal{K} \times \mathcal{M}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k(m)$*. The decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k, c) \in \mathcal{K} \times \mathcal{C}$ *and outputs a string* $m \leftarrow \mathsf{Dec}_k(c)$ *which is either a string in* $\mathcal{M}$ *or the symbol "*$\perp$*" (invalid).*
We require that the algorithm Enc *and* Dec *are the inverse of each other, that is:*
Correctness: *if* $c \leftarrow \mathsf{Enc}_k(m)$ *then* $m \leftarrow \mathsf{Dec}_k(c)$*,*
Tidiness: *if* $m \leftarrow \mathsf{Dec}_k(c)$ *with* $m \neq \perp$ *then,* $c \in \{c \leftarrow \mathsf{Enc}_k(m)\}$*, that is, c may be obtained as encryption of m using key k.*
If $m \leftarrow \mathsf{Dec}_k(c)$ *with* $m \neq \perp$ *we say that c is* valid*, otherwise it is* invalid*.*

But usually encryption algorithms are deterministic, thus, we can see the Enc algorithm of the previous definition, as composed by two elements: a standard deterministic encryption algorithm $\mathsf{Enc} : \mathcal{K} \times \mathcal{R} \times \mathcal{M} \longmapsto \mathcal{C}$ and an algorithm which provides to the previous one the randomness $r \in \mathcal{R}$ which is needed. Thus, if we want to highlight this and "outsource" the random generator, we have the following definition:

Definition 9. *An* IV-based authenticated encryption scheme *(*ivAE*) is a triple of algorithm* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a deterministic algorithm which takes as input the tuple*

$(k, iv, m) \in \mathcal{K} \times \mathcal{IV} \times \mathcal{M}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k(iv, m)$. *The decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k, c) \in \mathcal{K} \times \mathcal{C}$ *and outputs a string* $m \leftarrow \mathsf{Dec}_k(c)$ *which is either a string in* $\mathcal{M}$ *or the symbol "*$\perp$*" (invalid).*
We require that the algorithm Enc *and* Dec *are the inverse of each other, that is:*
Correctness: *if* $c \leftarrow \mathsf{Enc}_k(iv, m)$, *then,* $m \leftarrow \mathsf{Dec}_k(c)$,
Tidiness: *if* $m \leftarrow \mathsf{Dec}_k(c)$ *then, there exists an* $iv \in \mathcal{IV}$ *s.t.* $c \leftarrow \mathsf{Enc}_k(iv, m)$.
If $m \leftarrow \mathsf{Dec}_k(c)$ *with* $m \neq \perp$ *we say that* c *is* valid, *otherwise it is* invalid.

This definition allows not to give as an output the IV, which, thus, may be considered secret. This has been used by Berti et al. [11] to have confidentiality in the presence of side-channel attacks and by Ashur et al. [3] to have the RUPAE security.
For ivAE, we cannot suppose that the IV is only not repeated, because, in such case, we loose the advantage of having it secret.

On the other hand, in the literature another notion is more used (for example, [31,35,8]). Its idea is to use only deterministic encryption algorithms and to provide them an additional input, which is transmitted with the ciphertext. This additional input should only not be repeated (thus, it is called *nonce*). This input makes the encryption still probabilistic (the security when this input is repeated is called *nonce-misuse* [35]).

Definition 10. *A* nonce-based authenticated encryption scheme *(*nAE*) is a triple of algorithm* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a deterministic algorithm which takes as input the tuple* $(k, n, m) \in \mathcal{K} \times \mathcal{N} \times \mathcal{M}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k(n, m)$. *The decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k, n, c) \in \mathcal{K} \times \mathcal{N} \times \mathcal{C}$ *and outputs a string* $m \leftarrow \mathsf{Dec}_k(n, c)$ *which is either a string in* $\mathcal{M}$ *or the symbol "*$\perp$*" (invalid).*
We require that the algorithm Enc *and* Dec *are the inverse of each other, that is:*
Correctness: *if* $c \leftarrow \mathsf{Enc}_k(n, m)$, *then,* $m \leftarrow \mathsf{Dec}_k(n, c)$,
Tidiness: *if* $m \leftarrow \mathsf{Dec}_k(n, c)$ *with* $m \neq \perp$ *then,* $c \leftarrow \mathsf{Enc}_k(n, m)$.
If $m \leftarrow \mathsf{Dec}_k(n, c)$, *with* $m \neq \perp$, *we say that* c *is* valid, *otherwise it is* invalid.

This notion has been already used also in the leakage resilient contest [33,13,4].

Bellare [6] propose a syntax, which is halfway between that proposed in Def. 9 and that proposed in Def. 10, where a nonce is used, but it is not an input needed by the decryption algorithm. This may prevent some attacks.

A.4 Other security notions for AE

For clarity, we start restating the security for AE schemes (which was stated in Def. 2):

Definition 11. *An authenticated encryption* AE *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-AE *secure if the following advantage*

$$\mathsf{Adv}^{\mathsf{AE}}_{\Pi,\mathsf{A}} := \left| \Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right] \right| \leq \epsilon$$

for any (q_E, q_D, t)*-adversary* A*, where the key* k *is picked uniformly at random, the algorithm* $\$(m)$ *answers a random string of length* $|c|$ *with* $c \leftarrow \mathsf{Enc}_k(m)$*, and* $\perp(\cdot)$ *is an algorithm which answers always* $\perp$ *("invalid"). The adversary* A *may ask* q_E *encryption queries (to the left oracle) and* q_D *decryption queries (to the right oracle). If he receives* c *as an answer of the first oracle, [that is,* $c \leftarrow \mathsf{Enc}_k(m)$ *(or* $c \leftarrow \$(m)$*)] he is not allowed to query the second oracle on input* c*.*

This definitions provides confidentiality and authenticity in the same time. In fact, it supposes that the adversary is never able to ask a decryption query with a fresh and valid ciphertext. Moreover, ciphertexts should be indistinguishable from random.

Similarly for what done for IV-based encryption schemes (see, for example [22,12]), we can define the security for ivAE schemes, modifying the previous security definition:

Definition 12. *An authenticated encryption* ivAE *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-ivAE *secure if the following advantage*

$$\mathsf{Adv}^{\mathsf{ivAE}}_{\Pi,\mathsf{A}} := \left| \Pr\left[\mathsf{A}^{\mathsf{Enc}^{\$}_k(\cdot),\mathsf{Dec}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right] \right| \leq \epsilon$$

for any (q_E, q_D, t)*-adversary* A*, where the key* k *is picked uniformly at random, the oracle* $\mathsf{Enc}^{\$}(\cdot)$*, first picks uniformly at random an* IV $iv \xleftarrow{\$} \mathcal{IV}$*, then, it computes* $\mathsf{Enc}_k(iv, m)$*, the algorithm* $\$(m)$ *answers a random string of length* $|c|$ *with* $c \leftarrow \mathsf{Enc}^{\$}_k(m)$*, and* $\perp(\cdot)$ *is an algorithm which answers always* $\perp$ *("invalid"). The adversary* A *may ask* q_E *encryption queries (to the left oracle) and* q_D *decryption queries (to the right oracle). If he receives* c *as an answer of the first oracle, [that is,* $c \leftarrow \mathsf{Enc}^{\$}_k(m)$ *(or* $c \leftarrow \$(m)$*)] he is not allowed to query the second oracle on input* c*.*

This definition does not assume that the adversary knows iv. There are two possibilities, each of them allowed by this definition: first, the iv is kept secret, second, it is given as part of the ciphertext. n both case ivAE security is achievable (in the second case, it is enough to observe that iv is random).

There is also a security notion for nAE schemes:

Definition 13. *An authenticated encryption* nAE *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-nAE *secure if the following advantage*

$$\mathsf{Adv}^{\mathsf{nAE}}_{\Pi,\mathsf{A}} := \left| \Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot,\cdot),\mathsf{Dec}_k(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right] \right| \leq \epsilon$$

for any (q_E, q_D, t)-adversary A, where the key k is picked uniformly at random, the algorithm $\$(n, m)$ answers a random string of length $|c|$ with $c \leftarrow \mathsf{Enc}_k(n, m)$, and $\perp(\cdot, \cdot)$ is an algorithm which answers always $\perp$ ("invalid"). The adversary A may ask q_E encryption queries (to the left oracle) and q_D decryption queries (to the right oracle). Moreover, he is not allowed to repeat the first input, the nonce, in different encryption queries. If he receives c as an answer of the first oracle, [that is, $c \leftarrow \mathsf{Enc}_k(n, m)$ (or $c \leftarrow \$(n, m)$)] he is not allowed to query the second oracle on input (n, c).

Eliminating the requirement on non-repeating nonces, we obtain misuse-resistant [35]:

Definition 14. *An authenticated encryption* mrAE *scheme $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ is (q_E, q_D, t, ϵ)-*mrAE *(*misuse resistant*) secure if the following advantage*

$$\mathsf{Adv}^{\mathsf{nAE}}_{\Pi,\mathsf{A}} := \left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot,\cdot),\mathsf{Dec}_k(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right]\right| \leq \epsilon$$

for any (q_E, q_D, t)-adversary A, where the key k is picked uniformly at random, the algorithm $\$(n, m)$ answers a random string of length $|c|$ with $c \leftarrow \mathsf{Enc}_k(n, m)$, and $\perp(\cdot, \cdot)$ is an algorithm which answers always $\perp$ ("invalid"). The adversary A may ask q_E encryption queries (to the left oracle) and q_D decryption queries (to the right oracle). If he receives c as an answer of the first oracle, [that is, $c \leftarrow \mathsf{Enc}_k(n, m)$ (or $c \leftarrow \$(n, m)$)] he is not allowed to query the second oracle on input (n, c).

There is a flourishing literature on misuse-resistant schemes [35,34].

A.5 Leakage

To be used the previous algorithms and primitives need to be implemented either on hardware or on software. But this implementations may leak physical information about their internal states and secrets (for example due to time [1], power consumption [27,24] or electromagnetic radiation [26]). These leakages have been used to break cryptographic schemes. Such attacks are called *side-channel attacks.*

Let the algorithm Alg be called on input $(x; k)$ where x is a public input and k a secret one. We model the leakage function as $\mathsf{L}_{\mathsf{Alg}}(x; k)$. This is called the *leakage function.*
First of all we observe that, to the best of our knowledge, we are not able to give an explicit formula for the leakage function. It is even possible that it may not be efficiently computable.
On the other hand, there are protection against such attacks. They are usually very expensive [18]. Our approach is the following:
First, we observe that usually an encryption or a decryption algorithm are usually divided in many steps. Each of these steps uses a different primitive, that

we call component.
Second, we consider the protection of each of these components. Instead. of protecting either everything well (very costly) or everything approximately (more efficient, but dangerous) we decide to have a primitive with a very well protected implementation and to have implementations of all components, including the one which has a well protected implementation, which are weakly protected [or, as we have already seen, for authenticity no protection at all]. This is called a *leveled implementation* (see Def.16 for a formal definition of it).
Moreover, for the very well protected implementation, we model it with the following definition:

Definition 15. *A keyed component (that is, a primitive which uses a key which is kept secret) has a* leak free implementation *if this primitive has an implementation which does not leak anything about its key.*

For simplicity, we call it *leak free.*
We observe that we do not assume anything for the leakage of the inputs and outputs of a leak free component. Some hypothesis about this leakage will be needed in the rest of the paper (see Def. 31).

The existence of leak free implementations is an open problem, but we have some primitives for which there exist very resistant to side-channel attacks implementations, for example block-ciphers implemented with an high order masking. On the other hand they are very expensive, this is why we try to use them as little as possible.
Thus, the AE mode we present, uses its primitives in a leveled implementation:

Definition 16. *An encryption (resp. decryption) algorithm* Enc *(resp.* Dec*) has a* leveled implementation *if* Enc *(resp.* Dec*) is implemented using a leak free component and other less protected components.*
Given $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$*, if both* Enc *and* Dec *have a leveled implementation, then, also scheme* Π *is has a* leveled implementation*.*

The goal of leveled implementations is to achieve provable security in a leakage scenario, being reasonably efficient.

For every security definition involving leakage, we precise the hypothesis that we assume on the leakage protection of the less protected component and of the inputs and outputs of the leak free component. We do this because in some definition we need nothing, while in others something more. This highlights also what happens if there is a security loss in the leakage protection of the less protected components, that is, what security property may be lost and what will be kept.

A.6 Ciphertext integrity with misuse and leakage in encryption and decryption

Berti et al. [13] introduce ciphertext integrity with misuse and leakage in encryption and decryption which is an evolution of ciphertext integrity with misuse and

leakage (in only encryption) introduced by Berti et al. [11]. Again we adapt it to the syntax used in Def. 1.

Definition 17. *An authenticated encryption (*AE*) scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-ciphertext integrity with misuse and leakage in encryption and decryption *(*CIML2*)-secure if the for all* (q, t)*-bounded adversaries, we have*

$$\Pr\left[\mathsf{CIML2}_{\Pi,\mathsf{L}_E,\mathsf{L}_D,\mathsf{A}} \Rightarrow 1\right] \leq \epsilon$$

where the CIML2 *experiment is defined in Tab. 2, where* r *is the randomness used by* Enc *in the encryption of the message* m*.*

$\mathsf{CIML2}_{\Pi,\mathsf{L}_E,\mathsf{L}_D,\mathsf{A}}(1^n)$ experiment	
Initialization: $k \leftarrow \mathcal{K}$ s.t. $\lvert k\rvert = n$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $c \leftarrow \mathsf{A}^{\mathsf{EncDL}_k(\cdot,\cdot),\mathsf{DecL}_k(\cdot)}$ If $c \in \mathcal{S}$ or $\perp = \mathsf{Dec}_k(c)$, Return 0 Return 1	Oracle $\mathsf{EncDL}_k(r, m)$: $c = \mathsf{EncD}_k(r, m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$ Return $(c, \mathsf{L}_E(r, m; k))$ Oracle $\mathsf{DecL}_k(c)$: Return $(\mathsf{Dec}_k(c), \mathsf{L}_D(c; k))$

Table 2. The CIML2 experiment. The corresponding definition in the blackbox model is *ciphertext integrity* [INT-CTXT] (with misuse) and is obtained from CIML2 imposing that $\mathsf{L}_E = \mathsf{L}_D = 0$ (i.e., there is no leakage).

Given the syntax for AE schemes we use (see Def. 1), we have to understand what means misuse. We suppose that the adversary has taken control of the random source (which may be, for example, a pseudorandom generator) which provides the randomness r. Thus, for us, misuse means that the adversary chooses and provides the randomness r to the encryption oracle (thus, it may be not picked randomly, but may be carefully chosen by the adversary and even repeated). Consequently, we denote the Enc algorithm with EncD to denote it.
Now we have left the problem of what is the leakage.
As in the other works [11,13,19,10], we use the unbounded leakage model, which is a very liberal model:

Definition 18. *In the* unbounded leakage model*, the leakage function* L *of an algorithm provides all internal states of the algorithm except those of the (eventual) leakfree components. In particular all keys, inputs, outputs and random coins of the unprotected components are given (thus, also all inputs and outputs (not the keys) of the leak free components).*

This model is very permissive [11], arguably one of the most permissive.

A.7 Release of unverified plaintexts (RUP)

We follow the approach of many other other works ([2,5,3]) in the RUP setting presenting *separated* AE *schemes*, but we have to adapt it to our AE syntax where

the randomness is not given as an input to the encryption oracle but picked by it. Moreover, only the ciphertext is transmitted, thus, the decryption algorithm uses no additional input.

Definition 19. *A* separated AE scheme *is a quadruple of algorithms* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{SDec}, \mathsf{SVer})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a probabilistic algorithm which takes as input the tuple* $(k, m) \in \mathcal{K} \times \mathcal{M}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k(m)$*. The (separated) decryption algorithm* SDec *is a deterministic algorithm which takes as input the tuple* $(k, c) \in \mathcal{K} \times \mathcal{C}$ *and outputs a string* $m \leftarrow \mathsf{Dec}_k(c)$ *while the (separated) verification algorithm* SVer *is a deterministic algorithm which takes as input the tuple* $(k, c) \in \mathcal{K} \times \mathcal{C}$ *and outputs either the symbol*
"$\top$ *" (valid) or "*$\perp$*'" (invalid).*

We require again the property of correctness and tidiness.

$$\textbf{Correctness}: \text{ if } c \leftarrow \mathsf{Enc}_k(m) \Rightarrow \mathsf{SDec}_k(c) = m \text{ and } \top \leftarrow \mathsf{SVer}_k(c)$$

$$\textbf{Tidiness}: \text{ if } \top \leftarrow \mathsf{SVer}_k(c) \Rightarrow \exists m \in \mathcal{M} \text{ s.t. } c \in \{\mathsf{Enc}_k(m)\}$$

$$\text{and if:} \top \leftarrow \mathsf{SVer}_k(c) \text{ and } m \leftarrow \mathsf{SDec}_k(c) \Rightarrow c \in \{\mathsf{Enc}_k(m)\}$$

Now we can define the RUPAE security definition:

Definition 20 ([5]). *A separated* AE *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{SDec}, \mathsf{SVer})$ *is* (q_E, q_D, t, ϵ)-RUPAE *secure if for any* (q_E, q_D, t)*-adversary* A *the following advantage*

$$\mathsf{Adv}^{\mathsf{RUPAE}}_{\Pi,\mathsf{A}} := \left| \Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$_E(\cdot),\$_D,\perp(\cdot)} \Rightarrow 1\right] \right| \leq \epsilon$$

where the key k *is picked uniformly at random, the algorithm* $\$_E(m)$ *answers a random string of length* $|c|$ *with* $c \leftarrow \mathsf{Enc}_k(m)$*, the algorithm* $\$_D(c)$ *outputs a random string of length* $|m|$ *with* $m \leftarrow \mathsf{SDec}_k(c)$ *and* $\perp(\cdot)$ *is an algorithm which answers always* $\perp$ *("invalid"). The adversary* A *is granted to* q_E *encryption query (to the left oracle) and* q_D *decryption query (to the right oracle). If he receives* c *as an answer of the first oracle, that is* $c \leftarrow \mathsf{Enc}_k(m)$ *(or* $c \leftarrow \$(m)$*) he is not allowed to query the second or third oracle on input* c*.*

For completeness we introduce the integrity definition in the RUP setting, adapted to our syntax: INT-RUP

Definition 21 ([2]). *A separated* AE *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{SDec}, \mathsf{SVer})$ *is* (q_E, q_D, t, ϵ)-INT-RUP*-secure if the for all* (q, t)*-bounded adversaries, we have*

$$\Pr\left[\mathsf{INT\text{-}RUP}_{\Pi,\mathsf{A}} \Rightarrow 1\right] \leq \epsilon$$

where the INT-RUP *experiment is defined in Tab. 3*

$\mathsf{INT\text{-}RUP}_{\Pi,\mathsf{A}}(1^n)$ experiment	
Initialization: $k \leftarrow \mathcal{K}$ s.t. $\lvert k\rvert = n$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $c \leftarrow \mathsf{Adv}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)}$ If $c \in \mathcal{S}$ or $\perp = \mathsf{SDec}_k(c)$, Return 0 Return 1	Oracle $\mathsf{Enc}_k(m)$: $c = \mathsf{Enc}_k(m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$ Return c Oracle $\mathsf{SDec}_k(c)$: Return $(\mathsf{SDec}_k(c))$ Oracle $\mathsf{SVer}_k(c)$: Return $(\mathsf{SVer}_k(c))$

Table 3. The INT-RUP experiment.

We often denote the previous probability with

$$\Pr[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)} \text{ wins } \mathsf{INT\text{-}RUP}]$$

Although it may appear that CIML2 security is stronger than INT-RUP one, they are not comparable. In fact, it is possible that the plaintext may not be recomputed until the validity of the ciphertext has been established. Thus, it may not be leaked.
Instead, if in decryption the plaintext is recomputed and, thus, leaked, as it is the case in this paper, the CIML2 security is stronger than the INT-RUP one. In fact, in addition with the plaintext the CIML2 adversary receives all internal states of the non leak free components and, in our syntax, he may also choose the randomness used for encryption queries.

A.8 Confidentiality with leakage in both encryption and decryption (CPAL2)

In this section we define confidentiality in the presence of leakage.
Pereira et al. [33] give the first definitions, while Guo et al. [19] do a detailed study of all the possible definitions.

We want to model the ability of an adversary to break the confidentiality when he receives leakage from encryption and decryption queries. It seems hard, if not irrealistic, to have the indistinguishability from a random string with leakage (it may be difficult to compute a leakage for the random string which is indistinguishable from a real execution of the algorithm [33]). Thus, we model confidentiality as the probability that an adversary is able to distinguish the encryption of two challenge plaintexts $m^{*,0}$ and $m^{*,1}$ of the same length, having also access to the leakage of their computations (as done by Pereira et al. [33]). Thus, we use the CPAL2 (chosen plaintext security with leakage in both encryption and decryption) and CCAL2 (chosen ciphertext security with leakage in both encryption and decryption) experiments, introduced by Guo et al. [19] (see Tab. 4).

We define the $\mathsf{PrivKey}^{\mathsf{CPAL2}}$ and $\mathsf{PrivKey}^{\mathsf{CCAL2}}$ games in Tab. 4. The difference between the $\mathsf{PrivKey}^{\mathsf{CPAL2}}$ and $\mathsf{PrivKey}^{\mathsf{CCAL2}}$ game is that in the latter one the adversary has access to a decryption oracle:

$\mathsf{PrivKey}^{\mathsf{CPAL2}}_{\mathsf{Enc},\mathsf{L}_E,\mathsf{L}_D,\mathsf{L}^u_E,\mathsf{A}}(1^n)$ and $\mathsf{PrivKey}^{\mathsf{CCAL2}}_{\mathsf{Enc},\mathsf{L}_E,\mathsf{L}_D,\mathsf{L}^u_E,\mathsf{L}^u_D\mathsf{A}}(1^n)$ experiments	
Initialization: $k \leftarrow \mathcal{K}$ s.t. $\lvert k\rvert = n$, $b \xleftarrow{\$} \{0,1\}$ Challenge output : $(m^{*,0}, m^{*,1}) \leftarrow \mathsf{A}^{\mathsf{EncL}_k(\cdot)}$ If $\lvert m^{*,0}\rvert \neq \lvert m^{*,1}\rvert$ Return 0; Else $(c^*, \mathsf{L}_E(m^{*,b}; k), \mathsf{L}_D(c^*; k)) \leftarrow \mathsf{Encch}_k(m^{*,0}, m^{*,1})$ Finalization: $b' \leftarrow \mathsf{A}^{\mathsf{EncL}_k(\cdot),\mathsf{DecL}_k(\cdot)}(c^*, \mathsf{L}_E(m_b; k), \mathsf{L}_D(c^*; k))$ If $b = b'$, Return 1 Return 0	Oracle $\mathsf{EncL}_k(r, m)$: $c = \mathsf{Enc}_k(m)$ Return $(c, \mathsf{L}_E(m; k))$ Oracle $\mathsf{Encch}_k(m^{*,0}, m^{*,1})$: $c^* \leftarrow \mathsf{Enc}_k(m^{*,b})$ Return $(c^*, \mathsf{L}_E(m_b; k), \mathsf{L}_D(c^*; k))$ Oracle $\mathsf{DecL}_k(c)$: Return $(\mathsf{Dec}_k(c), \mathsf{L}_D(c; k))$

Table 4. The $\mathsf{PrivKey}^{\mathsf{CPAL2}}$ and $\mathsf{PrivKey}^{\mathsf{CCAL2}}$ experiment. In the $\mathsf{PrivKey}^{\mathsf{CPAL2}}$ the adversary has not access to the $\mathsf{DecL}_k(\cdot)$ oracle

Definition 22. *An authenticated encryption* AE *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage* $\mathsf{L} = (\mathsf{L}_E, \mathsf{L}_D)$ *is* (q_E, t, ϵ)-CPAL2 *(*chosen plaintext attack secure in the presence of leakage in encryption and in encryption of the challenge plaintext and in decryption of the challenge ciphertext*) [resp.* (q_E, q_D, t, ϵ)-CCAL2 *(*chosen ciphertext attack secure in the presence of leakage in both encryption and decryption, and in encryption of the challenge plaintext and in decryption of the challenge ciphertext*)], if for every* (q_E, t) *[resp.* (q_E, q_D, t)*]-bounded adversary* A*, we have:*

$$\Pr\left[\mathsf{PrivKey}^{\mathsf{CPAL2}}_{\mathsf{A},\Pi,\mathsf{L}} = 1\right] \leq \frac{1}{2} + \epsilon$$

$$\left[\ \textit{resp.}\ \Pr\left[\mathsf{PrivKey}^{\mathsf{CCAL2}}_{\mathsf{A},\Pi,\mathsf{L}} = 1\right] \leq \frac{1}{2} + \epsilon\right]$$

where the adversary is granted at most most q_E *encryption queries [resp. and* q_D *decryption queries] and he runs in time bounded by* t*.*
If $q_E = q_D = 0$*, that is, the adversary does only the challenge query, the scheme is called* (t, ϵ) *(*eavesdropper secure in the presence of leakage in encryption and decryption*) (*EavL2*) secure. If also* $\mathsf{L}_D = 0$ *(that is, there is no leakage in decryption), the scheme is called* (t, ϵ) *(*eavesdropper secure in the presence of leakage in encryption*) (*EavL*) secure.*

(The 2 of CPAL2, etc., is to remind that the adversary A has access to the leakage of the decryption of the challenge query[ies]).
Guo et al. [19] allowed the adversary to ask multiple challenge queries, here, for

simplicity we consider only one.

Leakage model (the PSV one) We are left with the problem of what leakage model we use. Since, we do not introduce any novelty in the encryption part, we decide to reuse the model presented by Standaert et al. [37]: the *simulatability*. (At the end we have to introduce some technical addition, whose necessity will be clear in Sec. D.4.

The notion of simulatability is based on the $\mathsf{q\text{-}sim}$ game presented in Tab. 5. In this game we measure the capability of a simulator to produces leakages which seem consistent with the input and outputs of a PRF, without knowing the key used by the PRF. Thus, we can define:

Game $\mathsf{q\text{-}sim}(\mathsf{A}, \mathsf{PRF}, \mathsf{L}, \mathcal{S}, b)$ [37, Section 2.1].		
The challenger selects two random keys $k, k^* \xleftarrow{\$} \mathcal{K}$. *The output of the game is a bit* b' *computed by* A^{L} *based on the challenger responses to a total of at most* q *adversarial queries of the following type:*		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}(x)$	$\mathsf{E}_k(x)$, $\mathsf{L}(k, x)$	$\mathsf{E}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{E}_k(x))$
and one query of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}\text{-}\mathcal{S}(z, x)$	$\mathcal{S}^{\mathsf{L}}(z, x, k)$	$\mathcal{S}^{\mathsf{L}}(z, x, k^*)$

Table 5. The $\mathsf{q\text{-}sim}$ experiment of Standaert et al. [37].

Definition 23. [q-simulatable leakages [37, Def. 1]] *Let* E *be a PRF having leakage function* L. *Then* E *has* $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_{\mathsf{A}}, t_{\mathsf{A}}, \epsilon_{\mathsf{q\text{-}sim}})$ *q-simulatable leakages if there is a* $(q_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded simulator* $\mathcal{S}^{\mathsf{L}}$ *such that, for every* (q_{L}, t)*-bounded adversary* A^{L}, *we have*

$$|\Pr[\mathsf{q\text{-}sim}(\mathsf{A}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 1) = 1] - \Pr[\mathsf{q\text{-}sim} \quad (\mathsf{A}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1]| \leq \epsilon_{\mathsf{q\text{-}sim}}.$$

We observe that A is granted q_{L} queries to the leakage oracle. This queries are different from the queries done by the challenger. In fact for the queries done by A, he chooses the key and the plaintext, thus, they are intended to profile the leakage of E.

This assumption is useful to reduce the leakage security of the whole encryption to the leakage security of the encryption of a single block. What is the leakage security of a single block is still an open problem, but it may be studied and evaluated much more easily [33].

The EavL [33,11] and $\mathsf{EavL2}$ [13] security of PSV has already been established in the literature. It was possible to reduce the security of the whole scheme to

the security of an adversary who has only access to an ideal encryption of a single n bit (a block) message, called $\mathsf{PSV}s^I$ (see Tab. 7).

Other leakage assumptions In addition, with respect to the PSV model [33], we need some additional hypothesis on the leakage of E and F^*. The reasons, why we need them, are discussed in Sect. D.4, in particular in Lemma 4.

Definition 24. *The leak free implementation of keyed primitive has $(q, q_{\mathcal{S}'}, t, t_{\mathcal{S}'})$-indistinguishable leakage if for any (q, t) adversary, there exists a $(q_{\mathcal{S}'}, t_{\mathcal{S}'})$-simulator such that the leakage $\mathsf{L}_{\mathsf{F}^*}(x, y; k)$ of the computation $z \leftarrow \mathsf{F}^*_k(x, y)$ is indistinguishable from the simulated leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(x, y, z, k^*)$ for a random key k^*.*

Since, by hypothesis, the leak free component hides the key it uses, the previous definition can be assumed given the leak free hypothesis, although it is very strong. In particular, we suppose that the leakage of F^* depends only on its inputs and outputs. We may also suppose that the difference is bounded by ϵ, but we choose to suppose that they are completely indistinguishable, due to the hypothesis that F^* has a leak free implementation.

Regarding the leakage of k_0, when it is picked uniformly at random, we suppose that its leakage is given by a simulator $\mathcal{S}_\$$ which uses as only input k_0.

Now, we can modify the $\mathsf{q\text{-}sim}$ game in order to consider the additional source of leakage for k_0, obtaining the $\mathsf{q\text{-}sim}'$ game:

Game $\mathsf{q\text{-}sim}'(\mathsf{A}, \mathsf{PRF}, \mathsf{L}, \mathcal{S}, \mathcal{S}_{\mathsf{F}^*}, b)$		
The challenger selects three random keys $k, k^*, k^+ \xleftarrow{\$} \mathcal{K}$ and a random value $w \xleftarrow{\$} \{0,1\}^n$. The output of the game is a bit b' computed by A^{L} based on the challenger responses to a total of at most q adversarial queries of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{E}, \mathsf{L}_{\mathsf{E}}$ \ $\mathsf{E}, \mathcal{S}^{\mathsf{L}}(x)$	$\mathsf{E}_k(x)$, $\mathsf{L}(k, x)$	$\mathsf{E}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{E}_k(x))$
and one query of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}\text{-}\mathcal{S}'()$	$\mathsf{L}_\$(k)$	$\mathsf{L}_\$(k^*)$
and one query of the following type:		
$\mathsf{Key\text{-}Send}(h)$	$\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(h, k, k^+, w)$	$\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(h, k^*, k^+, w)$

Table 6. The $\mathsf{q\text{-}sim}'$ experiment used for the first block. Note that A may choose h after having received the answer of the previous queries.

Definition 25. [q-simulatable leakages'] *Let E be a PRF having leakage function L and let F^* be a STPRP having $(q_{\mathcal{S}'}, t_{\mathcal{S}'})$-indistinguishable leakage (see*

Def. 31). Then E *has* $(q_{\mathsf{L}}, q_{\mathcal{S}}, q_{\mathcal{S}'}, t, t_{\mathcal{S}}, t_{\mathcal{S}'}, \epsilon_{\mathsf{q\text{-}sim}})$ *q-simulatable' leakage if there is a* $(q_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded simulator* $\mathcal{S}^{\mathsf{L}}$ *such that, for every* (q_{L}, t)*-bounded adversary* A^{L}*, we have*

$$|\Pr[\mathsf{q\text{-}sim}'(\mathsf{A}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 1) = 1] - \Pr[\mathsf{q\text{-}sim}'(\mathsf{A}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1]| \leq \epsilon_{\mathsf{q\text{-}sim}}.$$

(The q-simulatability' assumption is clearly the q-simulatability assumption [Def. 23 or [33]] with the two modifications: the key k is not generated via E but picked uniformly at random and it is also encrypted using F^* with a tweak h chosen by the adversary A).

Other leakage model Yu et al. [39], reused by Berti et al. [10] model the leakage function of the PRF E as $\mathsf{L} = (\mathsf{L}_{\mathsf{E}}^{in}, \mathsf{L}_{\mathsf{E}}^{out})$, where $\mathsf{L}_{\mathsf{E}}^{in}$ is the leakage given by the first part of the computation of E, while $\mathsf{L}_{\mathsf{E}}^{out}$ by the last part of the computation. In particular, they suppose that $\mathsf{L}_{\mathsf{E}}^{in}$ is a function of the inputs and the key of the PRF E, that is, if $y = \mathsf{E}_k(x)$, then $\mathsf{L}_{\mathsf{E}}^{in}(x;k) := \mathsf{L}^{in}(x;k)$, and $\mathsf{L}_{\mathsf{E}}^{out}$ is a function of the outputs and the key, that is, $\mathsf{L}_{\mathsf{E}}^{out}(x;k) := \mathsf{L}^{out}(y;k)$.
This allows to avoid unrealistic attacks [for example, consider a stateful PRG based on E, (this PRG provides a key stream), which rekeys at every round of its execution, we want to avoid the possibility of a completely unrealistic attack where the leakage of a round may give information about the key used 10 rounds after].
Since, in their hypothesis the PRG which PSV is based on, is leakage resilient, also CONCRETE should be leakage resilient in that model. But, for space constraints, we do not study this in detail and we leave it as an open problem.

B Extension to Associated Data

B.1 Syntax and security

It may happen that there are some data which need to be authenticated, but not encrypt (for example, an header containing metadata like the sender, the receiver, the time). In such situation in the nonce case, syntactically the scheme is:

Definition 26. *[[31]] A* nonce-based authenticated encryption with associated data scheme *(*nAEAD*) is a triple of algorithm* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a deterministic algorithm which takes as input the tuple* $(k, n, a, m) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k(n, a, m)$*. The decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k, n, a, c) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C}$ *and outputs a string* $m \leftarrow \mathsf{Dec}_k(n, a, c)$ *which is either a string in* $\mathcal{M}$ *or the symbol "*$\perp$*" (invalid). We require that the algorithm* Enc *and* Dec *are the inverse of each other, that is:*
Correctness: *if* $c \leftarrow \mathsf{Enc}_k(n, a, m)$*, then,* $m \leftarrow \mathsf{Dec}_k(n, a, c)$*,*
Tidiness: *if* $m \leftarrow \mathsf{Dec}_k(n, a, c)$ *with* $m \neq \perp$ *then,* $c \leftarrow \mathsf{Enc}_k(n, a, m)$*.*

If $m \leftarrow \mathsf{Dec}_k(n, a, c)$*, with* $m \neq \perp$*, we say that* c *is* valid*, otherwise it is* invalid*.*

For such a scheme, its security is defined as follow:

Definition 27. *An authenticated encryption with associated data* nAEAD *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-nAE *secure if the following advantage*

$$\mathsf{Adv}^{\mathsf{nAEAD}}_{\Pi,\mathsf{A}} := \left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot,\cdot,\cdot),\mathsf{Dec}_k(\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1\right]\right| \leq \epsilon$$

for any (q_E, q_D, t)*-adversary* A*, where the key* k *is picked uniformly at random, the algorithm* $\$(n, a, m)$ *answers a random string of length* $|c|$ *with* $c \leftarrow \mathsf{Enc}_k(n, a, m)$*, and* $\perp(\cdot, \cdot, \cdot)$ *is an algorithm which answers always* $\perp$ *("invalid"). The adversary* A *may ask* q_E *encryption queries (to the left oracle) and* q_D *decryption queries (to the right oracle). Moreover, he is not allowed to repeat the first input, the nonce, in different encryption queries. If he receives* c *as an answer of the first oracle, [that is,* $c \leftarrow \mathsf{Enc}_k(n, a, m)$ *(or* $c \leftarrow \$(n, a, m)$*)] he is not allowed to query the second oracle on input* (n, a, c)*.*

Now, we have to adapt these definitions to our environment, where, instead of a nonce, a random value is picked and kept secret. First, we define the syntax:

Definition 28. *An* authenticated encryption with associated data scheme *(*AEAD*) is a triple of algorithm* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a probabilistic algorithm which takes as input the tuple* $(k, a, m) \in \mathcal{K} \times \mathcal{AD} \times \mathcal{M}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k(a, m)$*. The decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k, a, c) \in \mathcal{K} \times \mathcal{AD} \times \mathcal{C}$ *and outputs a string* $m \leftarrow \mathsf{Dec}_k(a, c)$ *which is either a string in* $\mathcal{M}$ *or the symbol "*$\perp$*"* *(invalid).*
We require that the algorithm Enc *and* Dec *are the inverse of each other, that is:*
Correctness: *if* $c \leftarrow \mathsf{Enc}_k(a, m)$*, then,* $m \leftarrow \mathsf{Dec}_k(a, c)$*,*
Tidiness: *if* $m \leftarrow \mathsf{Dec}_k(a, c)$ *with* $m \neq \perp$ *then,* $c \leftarrow \mathsf{Enc}_k(a, m)$*.*
If $m \leftarrow \mathsf{Dec}_k(a, c)$*, with* $m \neq \perp$*, we say that* c *is* valid*, otherwise it is* invalid*.*

Its security is provided by the following definition:

Definition 29. *An authenticated encryption with associated data* nAEAD *scheme* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-nAE *secure if the following advantage*

$$\mathsf{Adv}^{\mathsf{AEAD}}_{\Pi,\mathsf{A}} := \left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot,\cdot),\mathsf{Dec}_k(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right]\right| \leq \epsilon$$

for any (q_E, q_D, t)*-adversary* A*, where the key* k *is picked uniformly at random, the algorithm* $\$(a, m)$ *answers a random string of length* $|c|$ *with* $c \leftarrow \mathsf{Enc}_k(a, m)$*, and* $\perp(\cdot, \cdot)$ *is an algorithm which answers always* $\perp$ *("invalid"). The adversary* A *may ask* q_E *encryption queries (to the left oracle) and* q_D *decryption queries (to the right oracle). Moreover, if he receives* c *as an answer of the first oracle, [that is,* $c \leftarrow \mathsf{Enc}_k(a, m)$ *(or* $c \leftarrow \$(a, m)$*)] he is not allowed to query the second oracle on input* (a, c)*.*

which is Def. 27 adapted to the syntax introduced in Def. 28.

B.2 CONCRETE for associated data

It is easy to modify CONCRETE to consider also authenticated data (AD). The first idea is that it is enough instead of computing $h = \mathsf{H}(c_0\|...\|c_l)$, we can compute $h = \mathsf{H}(a\|c_0\|...\|c_l)$. This works, if the hash function treats differently the associated data and the ciphertext; otherwise it may create problem: consider the AD a parsed in n-bit blocks $a = (a_1, ..., a_{l'})$, we have that $\mathsf{H}(a_1, ..., a_{l'}, c_0, c_1, ..., c_l) = \mathsf{H}(c'_0, ..., c'_{l'+l+1})$ with $c'_0 = a_1, ..., c'_{l'} = a_{l'}, c'_{l'+1} = c_0, ..., c'_{l'+l+1} = c_l$ (thus, we may have a collision on h simply moving some blocks from AD to ciphertext. It may be used to create a forgery.
Thus, we propose to modify the hash function, defining $\mathsf{H}'(a, m) :=$

- parse a in n-bit blocks, $a = (a_1, ..., a_{l'})$ (the last block is padded if necessary)
- parse c in n-bit blocks, $m = (c_0, ..., c_l)$ (again, the last block is padded if necessary)
- $\mathsf{H}'(a, m) := \mathsf{H}(a_1\|0\|a_2\|0\|...\|a_{l'}\|0\|c_0\|1\|c_1\|1\|...\|c_l\|1)$

(that is, we pad a 0 after every AD block and a 1 after every ciphertext block).

In this way, the previous attack is ruled out. (It is described in Fig. 1)

It may be proved that this variant has the same security as the original CONCRETE.

C Previous works

Notations Since we use the $\mathsf{PRF}\ \mathsf{E} : \mathcal{K}(= \{0,1\}^n) \times \{0,1\}^n \leftarrow \{0,1\}^n$ and the leakfree $\mathsf{STPRP}\ \mathsf{F}^* : \mathcal{K}(= \{0,1\}^n) \times \mathcal{TW}(= \{0,1\}^n) \times \{0,1\}^n \leftarrow \{0,1\}^n$, we call *block* a string of n bits. When we parse a message m in blocks, (that is, $m = (m_1, ..., m_l)$ with $m = m_1\|...\|m_l$, $|m_1| = ... = |m_{l-1}| = n$ and $m_l| \leq n$) the last block may not be full, that is, its length may be $\leq n$. Doing an abuse of notation we call it also a block. Let l be the number of blocks of the plaintext m.

There is an interesting line of works to achieve leakage-resilient authenticated encryption (AE), started at CCS 2015 by Pereira et al. ([33,11,13,19,10]). All this works assumes the existance of a very well-protected (*leak free*) component and others much less protected.
Assuming these hypothesis on the components, the key k of the scheme (usually called *master key*) *should only be used in the leak free components as a key.*

Their schemes are based on the *rekeying*, that is, every key used in the less protected component is used a few number of times in all the history of the game (typically 2). To do this they use, for example, the PRG (pseudorandom generator) of Standaert et al. [37], which, in every execution, computes a new internal key (called *ephemeral key*) and outputs a (pseudo)random value, thus, it may be used to generate a key stream.

All these schemes can be divided in 3 parts, see for example Fig. 3 showing it for EDT [13]:

Derivation of the first ephemeral key The first ephemeral key (which is usually denoted with either k_0 or k_1) is obtained using the master key k and the leak free component F^*. It may be obtained as $\mathsf{F}^*_k(x)$ where x may be picked uniformly at random ([33]), a nonce [13,19,10] or the tag ([11] to obtain nonce-misuse).

Encryption Usually, the encryption scheme introduced at CCS 2015 by Pereira et al. [33] is used. This scheme, presented in Fig. 2 uses the PRG of Standaert et al. [37]: starting from an ephemeral key k_1, a stream of pseudorandom bits is created which is XORed to the plaintext. It may happen that also the randomness may be encrypted (e.g. [11]). Guo et al. [19] use a more complicated scheme, which is substantially a double pass, in order to achieve nonce-misuse resistance. Berti et al. [10] used a variant of PSV using tweakable blockcipher to obtain multi-user security and beyond birthday security.

Authentication The tag τ is computed as $\tau = \mathsf{F}^*_k(h)$ where h is the hash of either the plaintext and the randomness ([11,13] or the ciphertext and the randomness ([13,19,10]). (A tweak may also be used).
To achieve CIML2, following the work of Berti et al. [13], the verification is not done recomputing the tag τ and checking if it is equal to the tag provided in the plaintext, but computing $h' = \mathsf{F}^{*,-1}_k(\tau)$ and checking if it is equal to the actual hash value (together with the pre-image resistance [13] of the hash function). That is, the decryption oracle, instead of saying "*the ciphertext is invalid since I know the right tag [which is τ' ($\neq \tau$)]*", (thus, risking to leak the right tag τ'), says "*the ciphertext is invalid since the tag τ is not the tag for this ciphertext, [but for those ciphertexts whose hash is h']*". Moreover, since h' is randomly picked (since F^* is a STPRP then, to obtain the security, it is enough to assume that it is difficult to find an hash pre-image for h' (range-oriented pre-image resistance).

It is not necessary that this the authentication is the last step, for example DTE, introduced by Berti et al. [11] starts with the authentication, then computes a first ephemeral key and finally encrypts the plaintext (it is based on the Tag-then-Encrypt paradigm).

Since we have isolated the encryption part, we want to give a syntax for it:

Definition 30. *A* rekeyed encryption scheme $\Pi = (\mathcal{K}, \mathsf{enc}, \mathsf{dec})$ *is a triple of algorithm s.t. the keyspace $\mathcal{K}$ is a nonempty set, the encryption algorithm* enc *is a deterministic algorithm which takes as input the tuple $(k, m) \in \mathcal{K} \times \mathcal{M}$ and outputs a string $c \leftarrow \mathsf{enc}_k(m)$. The decryption algorithm* dec *is a deterministic algorithm which takes as input the tuple $(k, c) \in \mathcal{K} \times \mathcal{C}$ and outputs a string $m \leftarrow \mathsf{dec}_k(c)$ which is either a string in $\mathcal{M}$ or the symbol "$\perp$" (invalid).*
We require that the algorithm enc *and* dec *are the inverse of each other, that is:*
Correctness: *if $c \leftarrow \mathsf{enc}_k(m)$ then $m \leftarrow \mathsf{Dec}_k(c)$,*

Tidiness: *if* $m \leftarrow \mathsf{dec}_k(c)$ *with* $m \neq \perp$ *then,* $c \in \{c \leftarrow \mathsf{enc}_k(m)\}$*, that is,* c *may be obtained as encryption of* m *using key* k*.*
If $m \leftarrow \mathsf{dec}_k(c)$ *with* $m \neq \perp$ *we say that* c *is* valid*, otherwise it is* invalid*.*

We observe that rekeyed encryption scheme are deterministic. This is not a problem, because, they should be used with a different key for every encryption query.

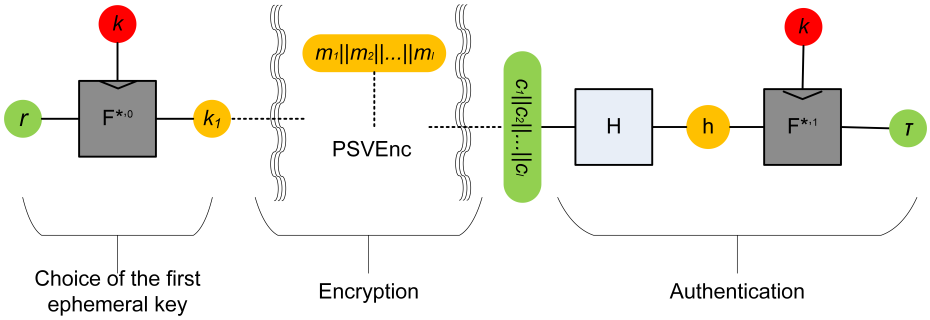

Fig. 3. The EDT mode separated in the three different phases. We use red for long term secret, orange for ephemeral one and green for outputs.

D Proofs

Notations We introduce some definitions and notations to make the proofs lighter. We use superscripts to indicate to which encryption and decryption query the message (or the ciphertext) is referred to. We use the set of index $\{1, ..., q_E\}$ for the encryption queries and $\{1, ..., q_D\}$ for decryption ones. [Formally, to make this two sets disjoint, we should write every index as (i, j) where j is either e or d, thus, for example, (i, e) indicates the ith encryption query. To make the notation lighter we omit the second component, because it is always clear if a query is an encryption or a decryption one].
The *transcripts* of the game consists of all queries made by the adversary to his oracles, the oracles's answers and the final output of the adversary. In a transcript every request is immediately followed by its answer.
Given a ciphertext $c = (c_0, ..., c_l, c_{l+1})$ we define the *partial ciphertext* as the vector $(c_0, ..., c_l)$, that is, the ciphertext without considering the block c_{l+1} encrypting the key.
An ephemeral key k_i is *fresh* if it should be picked uniformly at (pseudo)random.
During a game, a *previous* query is a query which appears before in the transcript of the game. It may be either an encryption or a decryption query.
During a game, given the partial ciphertext $(c_0, ..., c_l)$ we say that it is *fresh* if there exists no ciphertext c' obtained in a previous encryption query which has the same partial ciphertext [that is, $(c_0, ..., c_l) = (c'_0, ..., c'_l)$] and there is no ciphertext c'' asked for a previous decryption query which has the same partial

ciphertext [that is, $(c_0, ..., c_l) = (c''_0, ..., c''_l)$].
Given a partial ciphertext $(c_0, ..., c_l)$, we say that the block c_{l+1} is *fresh with respect to the partial ciphertext* $(c_0, ..., c_l)$ if there exists no ciphertext c' appearing in the transcription of the game s.t. $c' = (c_0, ..., c_l, c_{l+1})$ even if the partial ciphertext $(c_0, ..., c_l)$ is not fresh. That is, the block encrypting the key has never been paired with that particular partial ciphertext.
In the proofs we have also to consider the time needed to obtain a random block. We denote this time with $t_\$$. We suppose that every adversary has access to an own infinite random tape. Whenever he needs a random block, he reads n bits of this tape, thus $t_\$$ is the time needed to read n bits from this tape.

D.1 CIML2

Before proving the CIML2 security for CONCRETE, we have to define the leakage function in the unbounded model. We observe that $\mathsf{L}_E(r, m; k) := k_0 = r$, that is the leakage is the randomness r used by $\mathsf{Enc}_k(\cdot)$, because, from it all values can be recomputed apart k and c_{l+1} (but the latter is provided in the ciphertext). In particular all the inputs and outputs of every function can be recomputed by the adversary from r and the ciphertext. Moreover, also all the keys used in the non leak free components can be recomputed from r.
On the other hand, $\mathsf{L}_D(c; k) := k_0$, because from k_0 the adversary is able to recompute all values used in the decryption apart from k. Interestingly, when there is misuse, that is, the adversary has taken control of the random generator providing the randomness r, there is no leakage during encryption queries, since r is already known to the adversary, who has chosen it. We observe that if the adversary has obtained $c \leftarrow \mathsf{EncD}_k(r, m)$, the decryption leakage of c is $\mathsf{L}_D(c; k) := k_0 = r$, thus, there is no additional information that a CIML2 adversary can obtain via a leaking decryption query of a ciphertext he has already obtained as answer of an encryption query. Thus, to make the proof easier, we assume that the adversary does not ask the decryption of a ciphertext c if the adversary has obtained c as an answer of a encryption query.

Theorem 6. *Let* F^* *be a* $(Q+1, t+(Q+1)(t_\mathsf{H}+(2L+1)t_\mathsf{E}), \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudorandom permutation (*STPRP*), let* E *be a* $(2, t+(Q+1)(t_\mathsf{H}+(2L+1)t_\mathsf{E})+t_{\mathsf{f}(Q+1)}, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* $(0, t+(Q+1)(t_\mathsf{H}+(2L+1)t_\mathsf{E})+t_{\mathsf{f}(Q+1)}, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Then, the scheme* CONCRETE*, which encrypts messages which are at most* L*-block long, is* (q_E, q_D, t, ϵ)*-*CIML2 *secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{(q_D+1)(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}}$$

where $Q = q_E + q_D$, t_H *is the time necessary to evaluate the hash function* H*,* t_E *is the time to compute* $y = \mathsf{E}_k(x)$*,* $t_{\mathsf{f}(Q+1)}$ *is the time needed to lazy sample a tweakable random permutation at most* $Q+1$ *times.*

Proof. We uses a series of games. For simplicity, we call the decryption query induced by the output of A as the $q_D + 1$ decryption query.

Game 0 This is the real $\mathsf{CIML2}$ game where A attacks scheme $\mathsf{CONCRETE}$. Let E_0 be the event that A wins Game 0.

Game 1 In this game, we replace STPRP $\mathsf{F}_k^*(\cdot,\cdot)$ with the random tweakable permutation $\mathsf{f}^*(\cdot,\cdot)$. Let E_1 be the event that A wins Game 1.

Transition from Game 0 to Game 1 It is easy to build a $(Q{+}1, t{+}(Q{+}1)(t_{\mathsf{H}}{+}(2L+1)t_{\mathsf{E}}))$-$\mathsf{STPRP}$ adversary B whose STPRP advantage is $|\Pr[E_0] - \Pr[E_1]|$.

The STPRP adversary B The adversary B receives the key length n and has access to an oracle which is either implemented either via $\mathsf{F}_k^*(\cdot,\cdot)$ and $\mathsf{F}_k^{*,-1}(\cdot,\cdot)$ or via $\mathsf{f}^*(\cdot,\cdot)$ and $\mathsf{f}^{*,-1}(\cdot,\cdot)$. B has to distinguish the situations. In detail:
At the start of the game, B picks 2 constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function $\mathsf{H} : \{0,1\}^* \longmapsto \{0,1\}^n$ and sends to A (p_A, p_B, H). Moreover, B sets $\mathcal{S}$ as an empty set.
When A does an encryption query on input (r^i, m^i), with $m^i = (m_1^i, ..., m_{l^i}^i)$, B simply (1) sets $k_0^i := r^i$, then, (2) from the ephemeral key k_0^i, he computes $(c_0^i, ..., c_{l^i}^i)$, after that, (3) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i}^i)$ and (4) he queries his oracle on input $(h^i, k_0^i, +1)$ obtaining $c_{l^i+1}^i$, finally (5) B sets $\mathsf{L}_E(r^i, m^i; k) := r^i$ and answers A (c^i, r^i) with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$. Then, he updates the set $\mathcal{S}$ adding the ciphertext c^i. For every encryption query B does 1 oracle query, moreover, he evaluates E $2l^i + 1 \leq 2L + 1$ times and once the hash function H; thus, answering to A takes at most $(2L+1)t_{\mathsf{E}} + t_{\mathsf{H}}$ time.
When A makes a decryption query on input c^j with $c^j = (c_0^j, ..., c_{l^j}^j, c_{l^j+1}^j)$, B (1) computes the hash $h^j = \mathsf{H}(c_0^j, ..., c_{l^j}^j)$, (2) queries his oracle on input $(h^j, c_{l^j+1}^j, -1)$ obtaining k_0^j, (3) computes $\tilde{c}_0^j = \mathsf{E}_{k_0^j}(p_B)$, then (4) if $c_0^j \neq \tilde{c}_0^j$ he sets $m^j = \perp$, that is, he answers "invalid", otherwise, (5) from k_0^j, B is able to compute $m^j = (m_1^j, ..., m_{l^j}^j)$, finally, (6) B sets $\mathsf{L}_D(c^j, k) := k_0^j$ and he answers (m^j, k_0^j). For every decryption query B does 1 oracle query, moreover, he evaluates once the hash function H, once E if c^j is deemed invalid, otherwise $2l^j + 1 \leq 2L + 1$ times; thus, answering to A takes at most $(2L+1)t_{\mathsf{E}} + t_{\mathsf{H}}$ time.
When A outputs the challenge ciphertext $c = c^{q_D+1}$, B proceeds as for the others decryption queries. Thus, he uses for this decryption query again 1 oracle query and at most time $(2L+1)t_{\mathsf{E}} + t_{\mathsf{H}}$. If the decryption query c is valid and $c \notin \mathcal{S}$, B outputs 1, 0 otherwise.
Thus B runs in time bounded by $t + (Q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{E}})$ and does at most $Q + 1$ oracle queries.

Bounding $|\Pr[E_0] - \Pr[E_1]| \leq \epsilon_{\mathsf{STPRP}}$. Clearly if the oracle is implemented with $(\mathsf{F}_k^*(\cdot,\cdot), \mathsf{F}_k^{*,-1}(\cdot,\cdot))$ B is correctly simulating Game 0; otherwise, Game 1.

$$\text{Thus } \Pr[\mathsf{B}^{\mathsf{F}_k^*(\cdot,\cdot), \mathsf{F}_k^{*,-1}(\cdot,\cdot)} \Rightarrow 1] = \Pr[\mathsf{A} \text{ wins Game 0}] = \Pr[E_0]$$

$$\text{and } \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot,\cdot),\mathsf{f}^{*,-1}(\cdot,\cdot)} \Rightarrow 1] = \Pr[\mathsf{A} \text{ wins Game 1}] = \Pr[E_1].$$

Consequently

$$|\Pr[E_0] - \Pr[E_1]| = \left|\Pr[\mathsf{B}^{\mathsf{F}_k^*(\cdot,\cdot),\mathsf{F}_k^{*,-1}(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot,\cdot),\mathsf{f}^{*,-1}(\cdot,\cdot)} \Rightarrow 1]\right|$$

which is bounded by $\epsilon_{\mathsf{STPRP}}$ since $\mathsf{F}^*(\cdot,\cdot)$ is a a $(Q+1, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}}), \epsilon_{\mathsf{STPRP}})$-strong tweakable pseudorandom permutation (STPRP) and B is a $(Q+1, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})$-$\mathsf{STPRP}$ adversary.

Game 2 Game 2 is Game 1 where we suppose that all hash values h^i are different provided that their inputs are different. Let E_2 be the event that A wins Game 2.

Transition between Game 1 and Game 2 We introduce a failure event CR, so defined:

$$CR := \left\{ \begin{array}{c} \exists i, i' \in \{1,...,q_E\} \cup \{1,...,q_{D+1}\}, i \overset{\%}{\neq} i' \\ \text{s.t. } h^i = h^{i'} \text{ and } (c_0^i,...,c_{l^i}^i) \neq (c_0^{i'},...,c_{l^{i'}}^{i'}) \end{array} \right\}$$

(With the symbol $\overset{\%}{\neq}$ we mean that the inequality $i \neq i'$ *always* holds if one index is picked from $\{1,...,q_E\}$ and the other from $\{1,...,q_{D+1}\}$)
To compute the probability of event CR, which clearly consists on a collision for the hash function H, we build a collision resistant adversary C.

The $(0, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)})$ collision resistance adversary C The collision resistant adversary C wants to output a collision for the hash function H he has access to and he is based on the $\mathsf{CIML2}$ adversary A. To emulate either Game 1 or Game 2 for A, C simply picks two values $p_A, p_B \leftarrow \{0,1\}^n$ and a tweakable random permutation f^*. To make the adversary more efficient we allow him to lazy sample [8] the tweakable random permutation f^*. Then he behaves as adversary B emulating Game 0 (or 1) for A with two differences: first, to obtain c_{l+1}^i in encryption queries (step 4) and k_0^j in decryption queries (step 2), instead of querying his oracle and using its answers, C lazy samples $\mathsf{f}^*(\cdot,\cdot)$; second, he has a list $\mathcal{H}$ which he updates adding $(h^i, (c_0^i,...,c_{l^i}^i))$ every time he has to compute the hash function H (that is, he keeps track of all inputs and outputs of the hash function). At the end of the game, C looks up into his list $\mathcal{H}$: if he finds a collision, if it is the case, he outputs it, otherwise he outputs $(0,1)$. C does no query and he runs in time bounded by $t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)}$, where $t_{\mathsf{f}(Q)+1}$ is the time needed to lazy sample f^* $Q+1$ times.

Bounding $\Pr[CR]$ If event CR happens, clearly C wins because he has output a collision. Thus

$$\Pr[CR] \leq \Pr[\mathsf{C} \text{ produces a collision}] \leq \epsilon_{\mathsf{CR}}$$

since the hash function H is $(0, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)}, \epsilon_{\mathsf{CR}})$-collision resistant and C is a $(0, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)})$-adversary.

Bounding $|\Pr[E_1] - \Pr[E_2]|$ Since Game 1 and Game 2 are identical if event CR does not happen, then $\Pr[E_1] \leq \Pr_{[}\, E_2] + \Pr[CR] \leq \Pr[E_2] + \epsilon_{cr}$.

Game 3 Game 3 is Game 2 where we suppose that all fresh decryption queries are invalid. Let E_3 be the probability that A wins Game 3. Clearly $\Pr[E_3] = 0$.

Transition between Game 2 and 3 To bound the difference $|\Pr[E_2] - \Pr[E_3]|$ we build a sequence of $q_D + 2$ games Game 2^0, ..., Game 2^{q_D+1}.

Game 2^i Game 2^i is Game 2 where the first i decryption queries c^i, if they are fresh, are answered with $(\perp, k_0^i)$ with $k_0^i = \mathsf{L}_D(c^i)$. Let E_2^i be the event that adversary wins Game 2^i.
Clearly Game 2 is Game 2^0 and Game 3 is Game 2^{q_D+2}.

Transition between Game 2^{i-1} **and Game** 2^i We observe that the only difference between Game 2^{i-1} and Game 2^i is how the ith decryption query is treated. Consider the following event:

$$F_i := \{ \text{ the } i\text{th decryption query is valid and fresh}\}$$

If event F_i does not happen, Game 2^{i-1} and Game 2^i are indistinguishable since the answer to the ith decryption query is the same. Thus

$$\left|\Pr[E_2^{i-1}] - \Pr[E_2^i]\right| \leq \Pr[F_i].$$

Bounding $\Pr[F_i]$ Let $c^i = (c_0^i, c_1^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ be the ith decryption query. There are two possibilities:

F^1 The partial ciphertext $(c_0^i, ..., c_{l^i})$ is *fresh*,
F^2 The partial ciphertext $(c_0^i, ..., c_{l^i})$ is *not fresh*

Clearly every fresh ciphertext falls in exactly one of the previous case. We call event F_i^j event $F^j \cap F_i$.

Event F_i^1 Since the partial ciphertext $(c_0^i, ..., c_{l^i}^i)$ is fresh, then its hash h^i is fresh due to the fact that event CR has not happened. Since at least one of the input (the tweak in this case, which is equal to h^i) of the tweakable random permutation $\mathsf{f}^{*,-1}$ is fresh, then, $k_0^i := \mathsf{f}^{*,-1}(h^i, c_{l^i}^i)$ is picked uniformly at random via lazy sampling of the tweakable random permutation. Thus, event F_i^1 happens only if $\mathsf{E}_{k_0^i}(p_B) = c_0^i$ for a random key. This event is called CO^i (collision output). To compute $\Pr[CO^i]$ we introduce Game 3^i replacing $\mathsf{E}_{k_0^i}(\cdot)$ with the random function $\mathsf{f}^i(\cdot)$ in the computation of the ith decryption query.

Game 3^i Game 3^i is Game 2^i where we replace in the ith decryption query $\mathsf{E}_{k_0^i}$ with the random function f_0^i. We call event CO_2^i the event that $c_0^i = \tilde{c}_0^i$ in Game 2^i, and event CO_3^i the event that $c_0^i = \tilde{c}_0^i$ in Game 3^i.

Transition between Game 2^i and Game 3^i To do it we build a $(2, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)})$-$\mathsf{PRF}$ adversary D^i based on A.

The $(2, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)})$-$\mathsf{PRF}$ adversary D^i The PRF adversary D^i has access to an oracle which is implemented either with $\mathsf{E}_{k_0^i}(\cdot)$ where k_0^i is a key picked uniformly at random or with a random function $\mathsf{f}^i(\cdot)$. D^i has to distinguish the two situations. To emulate Game 2^i for A, D^i simply picks two values $p_A, p_B \leftarrow \{0,1\}^n$ and a tweakable random permutation f^*, which, for efficiency, he lazy samples. Then he behaves as adversary C emulating Game 1 for A, before the ith decryption query. When D^i receives the ith decryption query on input $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$, by hypothesis (F^1) the partial ciphertext is fresh, thus, its hash h^i is fresh $(\overline{CR})$. Then, D^i calls his oracle on input p_B, receiving y as an answer and he sets $\tilde{c}_0^i = y$. After that, he calls his oracle on input p_A receiving y' and he sets $k_1^i = y'$. Moreover, he picks a random key $\tilde{k}_0^i$ and he sets the leakage $\mathsf{L}_D(c^i; k) = \tilde{k}_0^i$. From now on, he behaves as C in Game 1. At the end of the game, if $\tilde{c}_0^i = c_0^i$, D^i outputs 1, otherwise he outputs 0. D^i does only two queries to his oracle and he runs in time bounded by

$$t+(q_E+i-1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)} \leq t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)}.$$

(Even if adversary D^i had not answered correctly to the ith decryption query, or if he had not simulated correctly the game after that query, this would not have created any problem since D^i's output does not depend on what the adversary A does after his ith decryption query.)
Moreover, with regard to the correctness of the simulation, we observe that, if the key k_0^i is a key which has already been used in the game as a key for E during a previous encryption or decryption query, it is not possible to replace $\mathsf{E}_{k_0^i}(\cdot)$ with a random function only in the ith decryption query. Apart from this case, the computation of $\tilde{c}_0^i$ is correctly simulated if the oracle is implemented with $\mathsf{f}^i(\cdot)$ for Game 3^i, while if the oracle is implemented with $\mathsf{E}_{k_0^i}(\cdot)$ for a random key k_0^i Game 2^i is correctly simulated by D^i. Thus, we define the event KC^i (key collision):

$$KC^i := \left\{ \begin{array}{l} \exists \text{ a previous encryption query } {}^{(r^j, m^j)} \text{ s.t. } \exists \lambda \text{ s.t. } k_0^i = k_\lambda^j \\ \text{or } \exists \text{ a previous decryption query } {}^{c^j} \end{array} \right\}$$

where among encryption queries j can run only among the encryption queries A has done before the ith decryption query, which are $(\leq q_E)$ and with the first $i-1$ decryption queries.

Bounding $\Pr[KC]^i$ Since k_0^i is randomly picked and since there are at most $i-1+q_E$ possible values that it cannot have, we can bound $\Pr[KC^i] \leq \frac{q_E(L+1)+(i-1)(L+1)}{2^n}$.

Bounding $\left|\Pr[CO_2^i] - \Pr[CO_3^i]\right|$ Since D^i is a $(2, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)})$-$\mathsf{PRF}$ adversary and $\mathsf{E}(\cdot)$ is a $(2, t+(Q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q+1)}, \epsilon_{\mathsf{PRF}})$-

PRF secure, thus

$$\left|\Pr[CO_2^i] - \Pr[CO_3^i]\right| = \left|\Pr[\mathsf{D}^{\mathsf{E}_{k_0^i}(\cdot)} \Rightarrow 1] - \Pr[\mathsf{D}^{\mathsf{f}^i(\cdot)} \Rightarrow 1]\right| \leq \epsilon_{\mathsf{PRF}}$$

Bounding $\Pr[CO_3^i]$ We can compute $\Pr[CO_3^i] = 2^{-n}$, since $\mathsf{f}_0^i(\cdot)$ is a random function, thus the probability that $\tilde{c}_0^i$ is equal to c_0^i is equal to $|\mathcal{T}|^{-1}$ with $\mathcal{T}$ the target space of the function $\mathsf{f}_0^i(\cdot)$.

We are finally able to bound $\Pr[F_i^1]$:
Bounding $\Pr[F_i^1]$ If D^i simulates correctly event F_i^1 happens iff event CO_i^2 happens, thus

$$\Pr[F_i^1] \leq \Pr[KC^i] + \Pr[CO_2^i] \leq \Pr[KC^i] + \Pr[CO_3^i] + \left|\Pr[CO_2^i] - \Pr[CO_3^i]\right|$$

Using the previous results, we obtain:

$$\Pr[F_i^1] \leq \Pr[KC^i] + \Pr[CO_3^i] + \left|\Pr[CO_2^i] - \Pr[CO_3^i]\right| \leq$$

$$\frac{q_E(L+1) + (i-1)(L+1)}{2^n} + 2^{-n} + \epsilon_{\mathsf{PRF}} =$$

$$\frac{q_E(L+1) + (i-1)(L+1) + 1}{2^n} + \epsilon_{\mathsf{PRF}}$$

Bounding $\Pr[F_i^2]$ Now the partial ciphertext $(c_0^i, ..., c_{l^i}^i)$ is not fresh and it has been obtained in encryption [1] and/or decryption queries. Now we can reuse everything we used to bound $\Pr[F_i^1]$ since c_{l^i+1} is fresh with respect to the partial ciphertext $(c_0^i, ..., c_{l^i}^i)$ and thus k_0^i is fresh. Consequently, it is uniformy picked at random. Still, $\Pr[KC^i] \leq \frac{q_E(L+1)+(i-1)(L+1)}{2^n}$. Consequently we can bound

$$\Pr[F_i^2] \leq \Pr[KC^i] + \Pr[CO_3^i] + \left|\Pr[CO_2^i] - \Pr[CO_3^i]\right| \leq$$

$$\frac{q_E(L+1) + (i-1)(L+1)}{2^n} + 2^{-n} + \epsilon_{\mathsf{PRF}} =$$

$$\frac{q_E(L+1) + (i-1)(L+1) + 1}{2^n} + \epsilon_{\mathsf{PRF}}$$

Bounding $\Pr[F_i]$ Since $\Pr[F_i] \leq \max\{\Pr[F_i^1], \Pr[F_i^2]\}$ we have bounded

$$\Pr[F_i] \leq \frac{q_E(L+1) + (i-1)(L+1) + 1}{2^n} + \epsilon_{\mathsf{PRF}}$$

[1] This may be done finding at most q_E different keys $k_0^1, ..., k_0^{q_E}$ s.t. $\mathsf{E}_{k_0^1}(p_B) = ... = \mathsf{E}_{k_0^{q_E}}(p_B)$ [very unlikely, but possible] and choosing $m_j^i = m_j^1 \oplus \mathsf{E}_{k_j^1}(p_B) \oplus \mathsf{E}_{k_j^i}(p_B)$. This can be done since the adversary chooses the k_0^is so he can anticipates all the values $\mathsf{E}_{k_j^i}(p_B)$ for every i and j. This is related to what we explain for the tidiness

Bounding $\left|\Pr[\mathsf{A}\ \textbf{wins Game}\ 2^{i-1}] - \Pr[\mathsf{A}\ \textbf{wins Game}\ 2^{i}]\right|$ We have already proved that $\left|\Pr[\mathsf{A}\ \text{wins Game}\ 2^{i-1}] - \Pr[\mathsf{A}\ \text{wins Game}\ 2^{i}]\right| \leq \Pr[F_i]$ thus we obtain that

$$\left|\Pr[\mathsf{A}\ \text{wins Game}\ 2^{i-1}] - \Pr[\mathsf{A}\ \text{wins Game}\ 2^{i}]\right| \leq \Pr[F_i] \leq$$

$$\frac{q_E(L+1)+(i-1)(L+1)}{2^n} + \epsilon_{\mathsf{PRF}}$$

Bounding $|\Pr[\mathsf{A}\ \textbf{wins Game}\ 3] - \Pr[\mathsf{A}\ \textbf{wins Game}\ 2]|$ Since Game 2 is Game 2^0 and Game 3 is Game 2^{q_D+1} we have:

$$|\Pr[\mathsf{A}\ \text{wins Game}\ 3] - \Pr[\mathsf{A}\ \text{wins Game}\ 2]| \leq$$

$$\sum_{i=1}^{q_D+1} \left|\Pr[\mathsf{A}\ \text{wins Game}\ 2^{i-1}] - \Pr[\mathsf{A}\ \text{wins Game}\ 2^{i}]\right| \leq$$

$$\sum_{i=1}^{q_D+1} \left(\frac{q_E(L+1)+(i-1)(L+1)+1}{2^n} + \epsilon_{\mathsf{PRF}}\right) =$$

$$\frac{(q_D+1)q_E(L+1)}{2^n} + \frac{\frac{q_D(q_D+1)(L+1)}{2}}{2^n} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}} =$$

$$\frac{(q_D+1)(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}}$$

Bounding $\Pr[E_3]$ Since all fresh decryption queries are deemed invalid in Game 3 there is no possibility that the adversary wins such Game thus $\Pr[E_3] = 0$.

We now are able to finally conclude the proof.
Bounding $\Pr[E_0]$ Using all the bounds about the event E_i $(i = 0, ..., 3)$ we obtain that

$$\Pr[E_0] \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{(q_D+1)(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}}$$

Observation on the bound Since we have supposed that all k_0 used in the decryption query are different (and different from all other ephemeral keys), the previous bound covers also the difference $\epsilon_{\mathsf{STPRP}} \setminus \epsilon_{\mathsf{tprf}}$ if we see F^* as a tweakable PRF, (that is, if $\mathsf{F}^*_k(\cdot,\cdot)$ is indistinguishable from a random function with the same signature).

Interestingly, we use the fact that E is a PRF to prove that the probability that $c_0 = \mathsf{E}_{k_0}(p_B)$ is negligible. This is a difference with respect to the other-works ([13,19,10]) where no hypothesis on E was used in the $\mathsf{CIML2}$ proof. The hypothesis that E is a PRF is too strong. We discuss this after the proof. On the other hand, since we need that E is a PRF is necessary to have the AE security, it is meaningful to prove the theorem with this hypothesis.
To make lighter the notation we do not use $\mathsf{EncD}_k(\cdot,\cdot)$, but we use $\mathsf{Enc}_k(\cdot,\cdot)$ since it is clear that for $\mathsf{CIML2}$ the encryption algorithm is deterministic since the randomness is provided by the adversary.

Observation on the proof The constraint, which we use for E, is stronger than what is in reality necessary, but, since, anyway we need that E is a $(2, t, \epsilon_{\mathsf{PRF}})$ for AE security (see Sec. 5.2) we assume anyway this hypothesis. On the other hand, we would have been able to prove the $\mathsf{CIML2}$ security, even if we had replaced E with the identity. The real security property that we need for the commitment $c_0 = c_0(k_0)$ is the following:

$$\forall c_0 \in \{0,1\}^n \ \Pr[\tilde{c}_0(k_0) = c_0 \text{ if } k_0 \xleftarrow{\$} \mathcal{K}'] \leq \epsilon_{\mathsf{COM}}$$

where $\tilde{c}_0(k_0)$ is the correct commitment for the ephemeral key k_0 which must be picked uniformly at random (in Thm. 1 we should replace ϵ_{PRF} with ϵ_{COM}). That is, given a commit c_0, the probability that it is correct for a random ephemeral key k_0, is negligible. In particular, this is true if the commitment is given by a PRF or by any injective function.
This also allows us to replace the term $\frac{(q_D+1)(L+1)(Q+q_E)}{2^{n+1}}$ with $\frac{(q_D+1)(Q+q_E)}{2^{n+1}}$, that is, we suppose only that all k_0 used in decryption queries are different.

Note on the ciphertext integrity ($\mathsf{INT\text{-}CTXT}$*)* We do not prove ciphertext integrity, because it is implied by the $\mathsf{CIML2}$-security. Anyway, the proof would be similar and the bound would be the same.

Tidiness We observe that our scheme is tidy. In fact, in the $\mathsf{CIML2}$ proof (Thm. 1), it may happen that, by pure chance the adversary is able to produce a ciphertext whose commitment c_0 is correct, thus, the ciphertext is valid. But, in such a case, this is the correct commitment for the ephemeral key k_0 obtained in the decryption. Thus, it is the ciphertext obtained when the randomness is $r = k_0$ and the message is the message obtained in decryption.

D.2 AE security

After having proved the authenticity, we want to prove the confidentiality, which should be based on the security of the PSV encryption scheme. We start studying confidentiality in the blackbox model, using the AE security definition, leaving the security in the presence of leakage to Sec. 5.4.
First, we prove the security for the scheme $\mathsf{CONCRETE}$. At the end, we give an idea the constraint we need on enc (which may replace the PSV) in order to have the AE security.

Theorem 7. *Let* F^* *be a* $(Q, t + Q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{E}}), \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudorandom permutation (*STPRP*), let* E *be a* $(2, t', \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* $(0, t+Q(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}})+t_{\mathsf{f}(Q)}, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Then, the scheme* $\mathsf{CONCRETE}$*, which encrypt messages which are at most* L*-block long, is* (q_E, q_D, t, ϵ)*-*AE *secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_D(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D}{2^n}$$
$$(q_E(L+1)+q_D)\epsilon_{\mathsf{PRF}} + \frac{q_E(L+1)[q_E(L+1)-1]}{2^{n+1}} + \frac{Q(Q-1)}{2^{n+1}}$$

where $Q = q_E + q_D$, t_{H} *is the time necessary to evaluate the hash function* H, t_{E} *is the time to compute* $y = \mathsf{E}_k(x)$, $t_{\mathsf{f}(Q)}$ *is the time needed to lazy sample a tweakable random permutation at most* Q *times,* $t_{\$}$ *the time necessary to obtain a random block and* $t' \leq$

$$t + t_{\mathsf{f}(q_E)} + q_E t_{\mathsf{H}} + 2 \max_{j=1,\ldots,L} [j t_{\$} + (L-j) t_{\mathsf{E}}] + 2(L+1) \max_{i=1,\ldots,q_E} [(i t_{\$} + (q_E - i) t_{\mathsf{E}}].$$

First, we observe that the scheme is $\mathsf{INT\text{-}CTXT}$ secure (since it is $\mathsf{CIML2}$ secure), then, we observe that all the ciphertext blocks can be replaced by random ones since either they are obtained via a STPRP with a different input (c_{l+1}) or via the PSV encryption scheme using a different key [since we can see the partial ciphertext $(c_0, ..., c_{l+1})$ as the PSV encryption of the message $0^n \| m$].

Proof. We define $\overline{\Pi} = (\mathcal{K}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ where $\overline{\mathsf{Enc}}$ is Enc where we replace the STPRP $\mathsf{F}_{k^*}(\cdot,\cdot)$ with the tweakable random permutation $\mathsf{f}^*(\cdot,\cdot)$ and we suppose that there are no collisions for the hash function. $\overline{\mathsf{Dec}}$ is modified accordingly. We observe that in reality the key of $\overline{\mathsf{Enc}_k}$ is f^*, so we denote it with $\overline{\mathsf{Enc}_{\mathsf{f}^*}}$.
As done by Berti et al. [12] we have that:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{AE}}_{\Pi,\mathsf{A}} := \left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| = \\
\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}_{\mathsf{f}^*}}(\cdot),\perp(\cdot)} \Rightarrow 1\right] + \right. \\
\left. \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}_{\mathsf{f}^*}}(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| \leq \\
\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}_{\mathsf{f}^*}}(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| + \\
\left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}_{\mathsf{f}^*}}(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| \leq \\
\left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}_k}(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| \\
\epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_D(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D}{2^n} + q_D \epsilon_{\mathsf{PRF}}
\end{aligned}$$

where the last inequality can be obtained by Thm. 1. In fact the proof of the aforementioned theorem can be used with the following changes:

- the adversary has no more a final decryption query, thus we replace $q_D + 1$ with q_D in the bound;
- The adversary A cannot pick the randomnesses rs, nor he may see them, thus all adversaries B, C, D used in the previous proof, when they have to simulate encryption queries for A they pick the randomnesses rs uniformly at random;
- since there is no leakage, the adversaries B, C, D used in the previous proof, have not to compute it and give it to A;

Since the oracle $\perp$ can be simulated correctly by anyone (it is enough to answer

$\perp$ to every decryption query without regarding it) we obtain

$$\left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}_{\mathsf{f}^*}}(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| = \left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}_{\mathsf{f}^*}}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot)} \Rightarrow 1\right]\right|$$

Now the proof is really similar to the proof done by Berti et al. [11]. We do it using again a series of games:

Game 0 It is the Game where the adversary is playing against $\overline{\mathsf{Enc}_{\mathsf{f}^*}}(\cdot)$. Let E_0 be the probability A outputs 1 at the end of this Game.

Game 1 It is Game 0 where there is no collision on the randomnesses rs. Thus, we introduce the event randomness collision

$$RC := \left\{\exists i, i' \in \{1, ..., q_E\},\ i \neq i' \text{ s.t. } r^i = r^{i'}\right\}$$

and we suppose it does not happen. This forces all the first ephemeral keys k_0^i to be different. Let E_1 be the probability that A outputs 1 at the end of this Game.

Bounding $\Pr[RC]$ Since the randomnesses rs are picked uniformly at random, we can use the birthday bound paradox([22]), thus, $\Pr[RC] \leq \frac{q_E(q_E-1)}{2^{n+1}}$ because the last randomness r^{q_E} cannot be equal to one of the others $q_E - 1$ randomnesses r^is.

Bounding $|\Pr[E_0] - \Pr[E_1]|$ Since Game 0 and Game 1 are identical if event RC does not happen, we have that

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[RC] \leq \frac{q_E(q_E-1)}{2^{n+1}}.$$

Game 2 It is Game 1 where we have replaced all ciphertext blocks of the partial ciphertexts with random blocks. Let E_2 be the probability A outputs 1 at the end of this Game.

Transition between Game 1 and Game 2 We use at most $q_E(L+2)$ Games $2^{I,J}$ with $I = 1, ..., q_E,\ J = -1, ..., l^i \leq L$.

Game $2^{I,J}$ it is Game 1 where for the first $I-1$ encryption queries we have replaced all blocks $(c_0^i, ..., c_{l^i}^i)$ with random values. Moreover for the Ith encryption query, we replaces the first J' blocks $(c_0^I, ..., c_{J'}^I)$ with random values, where $J' = min(J, l^I)$ [we do this, in order not to touch the last block $c_{l^I}^I$ which is obtained via f^*]. Moreover, at most $(I-1)(L+1)+J+1$ random keys $(k_0^i, ..., k_{l^i}^i$ and k_j^I, for $i = 1, ..., I-1$ and $j = 0, ..., J)$ are picked uniformly at random; in addition if $J \leq l^I$, a random key k_{J+1}^I is picked uniformly at random. From

k^I_{J+1} the remaining ciphertext blocks are computed. Let $E_2^{I,J}$ be the probability A outputs 1 at the end of the Game.

Transition between Game $2^{I,J-1}$ and Game $2^{I,J}$ If $J \geq l^I + 1$ Game $2^{I,J}$ and Game $2^{I,J+1}$ are identical, since we have replaced the same blocks of ciphertext with random and nothing more. Thus, $\Pr[E_2^{I,J-1}] = \Pr[E_2^{I,J}]$.
Otherwise, we build a $(2, t', \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ adversary $\mathsf{B}^{I,J}$ against the PRF $\mathsf{E}(\cdot)$ with $t' \leq$

$$t + t_{\mathsf{f}(q_E)} + q_E t_{\mathsf{H}} + 2 \max_{j=1,\ldots,L}[jt_{\$} + (L-j)t_{\mathsf{E}}] + 2(L+1) \max_{i=1,\ldots,q_E}[(it_{\$} + (q_E - i)t_{\mathsf{E}}].$$

The $(2,t')$-PRF adversary $\mathsf{B}^{I,J}$ adversary The PRF adversary $\mathsf{B}^{I,J}$ has access to an oracle which is implemented either with $\mathsf{E}_{k^I_J}(\cdot)$ or with a random function $\mathsf{f}^I_J(\cdot)$. He has to distinguish to the two situations. In detail:
First, the oracle, which adversary $\mathsf{B}^{I,J}$ faces, picks uniformly at random a key $k^I_J \leftarrow \{0,1\}^n$. Then, $\mathsf{B}^{I,J}$ picks 2 constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function $\mathsf{H} : \{0,1\}^* \longmapsto \{0,1\}^n$ and sends (p_A, p_B, H) to A. $\mathsf{B}^{I,J}$ also picks a tweakable random permutation $\mathsf{f}^*(\cdot,\cdot)$ which he lazy samples.
When A does an encryption query on input m^i, with $m^i = (m^i_1, ..., m^i_{l^i})$, if $i < I$, $\mathsf{B}^{I,J}$ simply (1) picks uniformly at random $r^i = k^i_0$, after that, (2) he picks uniformly at random $l^i + 1$ blocks $(c^i_0, ..., c^i_{l^i})$ with $|c^i_{l^i}| = |m^i_{l^i}|$, then, (3) he computes $h^i = \mathsf{H}(c^i_0 \| ... \| c^i_{l^i})$ and (4) he computes $c^i_{l^i+1} = \mathsf{f}^*(h^i, k^i_0)$ and answers c^i to A with $c^i = (c^i_0, ..., c^i_{l^i}, c^i_{l^i+1})$. Moreover (5) he picks l^i ephemeral keys $k^i_1, ..., k^i_{l^i}$ uniformly at random. For every of the first $I-1$ encryption queries adversary $\mathsf{B}^{I,J}$ lazy samples f^* one time, moreover, he picks uniformly at random $l^i+1 \leq L+1$ blocks a randmoness $r^i = k_0$ and at most L ephemeral keys $k^i_1, ..., k^i_l$ and he computes once the hash function H; thus, answering to A takes at most $t_{\mathsf{f}} + t_{\mathsf{H}} + 2(L+1)t_{\$}$ time.
When A does an encryption query on input m^i, with $m^i = (m^i_1, ..., m^i_{l^i})$, if $i = I$, if $J \geq l^I + 1$ adversary $\mathsf{B}^{I,J}$ encrypts m^I as he does for the first $I-1$ encryption queries; otherwise $\mathsf{B}^{I,J}$ simply (1) picks uniformly at random r^I and he sets $\tilde{k}^I_0 = r^I$, then, (2) for each $j < J$ he randomly picks the block c^I_j, (with $|c^I_{l^I}| = |m^I_{l^I}|$), after that, (3) for the Jth block he calls his oracle on input p_B receiving y^I_I and, if $J \neq l^I$ on input p_A obtaining k^I_{J+1} (if $J = l^I$, $\mathsf{B}^{I,J}$ calls his oracle only once on input P_B), then, (4) he computes $c^I_J := y^I_J \oplus m^J_I$ (if $J = l^I$, $c^I_J = \pi_{|m^I_I|}(y^I_J) \oplus m^I_J$, if $J = 0$, $c^I_J = y^I_J$) after that, (5) from k^I_{J+1} he is able to correctly compute $(c^I_{J+1}, ..., c^I_{l^I})$ then, (6) he computes $h^I = \mathsf{H}(c^I_0 \| ... \| c^I_{l^I})$ and finally (7) he computes $c^i_{l^I+1} = \mathsf{f}^*(h^I, \tilde{k}^I_0)$ and answers A c^I with $c^i = (c^I_0, ..., c^I_{l^I}, c^I_{l^I+1})$. Moreover (8) he picks $J-1$ ephemeral keys $k^I_1, ..., k^I_{J-1}$ at random. We observe that, although $k^I_0 \neq \tilde{k}^I_0$, that is the key $\mathsf{B}^{I,J}$ sends may be different from the key used in the oracle, this is not a problem since the tweak is fresh (due to event $\overline{CR}$), so the value $c^I_{l^I+1}$ is in both cases picked uniformly at random. For the Ith encryption query $\mathsf{B}^{I,J}$ lazy samples f^* one time, moreover, he picks uniformly at random J blocks, a randomness $r^I = k^I_0$ and $J-1$ ephemeral keys $k^I_1, ..., k^I_{J-1}$ evaluates $\mathsf{E}(\cdot)$ $2(l^I - J) \leq 2(L-J)$ times, and he computes once the hash function

H; thus, answering to A takes at most $t_{\mathsf{f}} + t_{\mathsf{H}} + 2Jt_{\$} + 2(L-J)t_{\mathsf{E}}$ which is bounded by $t_{\mathsf{f}} + t_{\mathsf{H}} + \max_{J=1,...,L}[2Jt_{\$} + 2(L-J)t_{\mathsf{E}}]$ time, with t_{f} the time necessary to lazy sample $\mathsf{f}^*(\cdot,\cdot)$. When A does an encryption query on input m^i, with $m^i = (m^i_1, ..., m^i_{l^i})$, if $i > I$, B simply (1) picks uniformly at random r^i and sets $k^i_0 := r^i$, then, (2) from the ephemeral key k^i_0, he computes $(c^i_0, ..., c^i_{l^i})$, after that, (3) he computes $h^i = \mathsf{H}(c^i_0 \| ... \| c^i_{l^i})$ and finally (4) he computes $c^i_{l^i+1} = \mathsf{f}^*(h^i, k^i_0)$ and (5) $\mathsf{B}^{I,J}$ answers c^i to A with $c^i = (c^i_0, ..., c^i_{l^i}, c^i_{l^i+1})$. For every of the last $q_E - I$ encryption queries $\mathsf{B}^{I,J}$ lazy samples f^* one time, moreover, he evaluates $\mathsf{E}(\cdot)$ $2(l^i+1) \leq 2(L+1)$ times, and he computes once the hash function H; thus, answering to A takes at most $t_{\mathsf{f}} + t_{\mathsf{H}} + 2(l^i+1)t_{\mathsf{E}}$ which is bounded by $t_{\mathsf{f}} + t_{\mathsf{H}} + 2(L+1)t_{\mathsf{E}}$ time. At the end of the Game A outputs a bit b, which is the bit output by $\mathsf{B}^{I,J}$. Thus in total the adversary $\mathsf{B}^{I,J}$ does 2 queries to his oracle, moreover he runs in time bounded by

$$t + t_{\mathsf{f}(q_E)} + q_E t_{\mathsf{H}} + 2(I-1)(L+1)t_{\$} + \max_{j=1,...,L}[2jt_{\$} + 2(L-j)t_{\mathsf{E}}] + (q_E - I - 1)2(L+1)t_{\mathsf{E}} \leq$$

$$t + t_{\mathsf{f}(q_E)} + q_E t_{\mathsf{H}} + \max_{j=1,...,L}[jt_{\$} + 2(L-j)t_{\mathsf{E}}] + 2(L+1)\max_{i=1,...,q_E}[(it_{\$} + (q_E - i)t_{\mathsf{E}}]$$

Ephemeral key collisions We want that the ephemeral keys are all different, in order to correctly simulate. We have already supposed that all the randomnesses rs are different (event RC). Now, we suppose that all rekeyed keys are different. To do this, it is enough to suppose that for the adversary $\mathsf{B}^{I,J}$, the ephemeral key k^I_{J+1} (if calculated) is different from all the other keys previously picked (that is, all the k^i_0 for $i = 1, ..., q_E$ and all k^I_js for $i < I,\ j = 1, ..., l^i$ (with $l^i \leq L$) and all k^I_j for $1 \leq j \leq J$). In fact, in these cases adversary $\mathsf{B}^{I,J}$ will have problems to correctly simulate Game $2^{I,J+1}$. Thus, we define the event of ephemeral key collision

$$EKC^{I,J} := \left\{ \begin{array}{l} k^I_{J+1} = k^i_j \text{ for an } i = 1, ..., I-1 \text{ or if } i = I \text{ and} \\ j = 0, ..., J-1 \text{ or if } i = I+1, ..., q_E \text{ and } j = 0 \end{array} \right\}$$

If the event $EKC^{I,J}$ happens in Game $2^{I,J}$ we abort. In Game $2^{I,J}$ since k^I_J is picked uniformly at random $\Pr[EKC^{I,J}] \leq \frac{(L+1)(I-1)+J+q_E-I}{2^n}$.

Bounding $|\Pr[E_2^{I,J-1}] - \Pr[E_2^{I,J}]|$ After having supposed that event $EKC^{I,J-1}$ does not happen, we observe that $\mathsf{B}^{I,J}$ simulates Game $2^{I,J-1}$ if the oracle is implemented with $\mathsf{E}_{k^I_J}(\cdot)$ otherwise Game $2^{I,J}$. Since $\mathsf{E}(\cdot)$ is a $(2, t', \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ we can bound if $J \neq 0$

$$|\Pr[E_2^{I,J-1}] - \Pr[E_2^{I+1,J}]| \leq \epsilon_{\mathsf{PRF}} + \Pr[EKC^{I,J}] \leq$$

$$\epsilon_{\mathsf{PRF}} + \frac{(I-1)(L+1) + J - 1 + q_E - I}{2^n}$$

while if $J = 0$, since event RC does not happen

$$|\Pr[E_2^{I,J-1}] - \Pr[E_2^{I+1,J}]| \leq \epsilon_{\mathsf{PRF}} + \Pr[EKC^{I,J}] \leq \epsilon_{\mathsf{PRF}} + 0$$

Bounding $|\Pr[E_1^{I,L}] - \Pr[E_1^{I+1,0}]|$ We observe that Game $2^{I,L}$ and Game $2^{I+1,0}$ are identical since in both games all the blocks of the partial ciphertexts of the first I encryption queries are random, and all the others are computed using $\overline{\mathsf{Enc}}(\cdot)$. Thus, $\Pr[E_2^{I,L}] = \Pr[E_2^{I+1,0}]$.

Bounding $|\Pr[E_1] - \Pr[E_2]|$ Using the previous 2 transitions and iterating them, we are finally able to bound $|\Pr[E_1] - \Pr[E_2]|$

$$
\begin{aligned}
&|\Pr[E_1] - \Pr[E_2]| \leq \\
&\quad \sum_{i=1}^{q_E} \left(\sum_{j=0}^{L} |\Pr[E_1^{i,j}] - \Pr[E_1^{i,j-1}]| \right) + \sum_{i=1}^{q_E - 1} |\Pr[E_1^{i,L}] - \Pr[E_1^{i+1,0}]| \leq \\
&\quad\quad \sum_{i=1}^{q_E} \left(\sum_{j=0}^{L} \epsilon_{\mathsf{PRF}} + \sum_{j=1}^{L} \frac{(i-1)(L+1) + j - 1 + q_E - i}{2^n} \right) + 0 \leq \\
&\quad\quad\quad q_E(L+1)\epsilon_{\mathsf{PRF}} + \sum_{i=1}^{q_E} \left(\sum_{j=1}^{L} \frac{(i-1)(L+1) + j - 1 + q_E - i}{2^n} \right) \leq \\
&\quad\quad\quad\quad q_E(L+1)\epsilon_{\mathsf{PRF}} + \frac{q_E(L+1)[q_E(L+1) - 1]}{2^{n+1}} - \frac{q_E(q_E - 1)}{2^{n+1}}
\end{aligned}
$$

where the last term is obtained observing that $\sum_{i=1}^{q_E} \left(\sum_{j=1}^{L} \frac{(L+1)(i-1)+j-1+q_E-i}{2^n} \right)$ is the sum of the first $q_E(L+1)-1$ natural numbers less the sum of the first $q_E - 1$ natural numbers.

Game 3 It is Game 2 where we have replaced all ciphertexts with random strings of the same length. Let E_3 be the event that the adversary outputs 1 at the end of this game. We observe $\Pr[E_3] = \Pr[\mathsf{A}^{\$(\cdot)} \Rightarrow 1]$.

Transition between Game 2 and Game 3 We have that the difference between Game 3 and Game 4 is between using a PRF instead of a STPRP. In fact, for every ciphertext c^i the first $l^i + 1$ blocks $(c_0^i, ..., c_{l^i}^i)$ are randomly picked, but the last ciphertext block $c_{l^i+1}^i$ is randomly picked according to a STPRP in Game 2, while in Game 3 is picked uniformly art random. Using the well known result about distinguishing a PRP from a PRF [8], we obtain that $\Pr[E_3] \leq \Pr[E_2] + \frac{Q(Q-1)}{2^{n+1}}$.

Bounding $\Pr[E_0]$ This conclude our proof, since using all the bounds already computed we obtain that

$$
\begin{aligned}
&|\Pr[E_0] - \Pr[E_1]| \leq \\
&\quad \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_D(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D}{2^n} + q_D\epsilon_{\mathsf{PRF}} + \frac{q_E(q_E-1)}{2^{n+1}} + \\
&\quad q_E(L+1)\epsilon_{\mathsf{PRF}} + \frac{q_E(L+1)[q_E(L+1)+1]}{2^{n+1}} - \frac{q_E(q_E-1)}{2^{n+1}} + \frac{Q(Q-1)}{2^{n+1}} \leq \\
&\quad \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_D(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D}{2^n} \\
&\quad (q_E(L+1)+q_D)\epsilon_{\mathsf{PRF}} + \frac{q_E(L+1)[q_E(L+1)-1]}{2^{n+1}} + \frac{Q(Q-1)}{2^{n+1}}
\end{aligned}
$$

Generic rekeyed encryption scheme If, we replace the PSV rekeyed encryption scheme with a generic one $\Pi = (\mathcal{K}', \mathsf{enc}, \mathsf{dec})$, the security we ask to Π in order to have the AE-security for scheme CONCRETE is the following:
for every $(0, t')$-adversary, the advantage

$$
\begin{aligned}
\mathsf{Adv}_{\Pi,\mathsf{A}} := \Big|\Pr[\mathsf{A}(c) \Rightarrow 1; k' \xleftarrow{\$} \mathcal{K}', m \leftarrow \mathsf{A}, c \leftarrow \mathsf{enc}_{k'}(m)] - \\
\Pr[\mathsf{A}(c) \Rightarrow 1; m \leftarrow \mathsf{A}, c \leftarrow \$(m)]\Big| \leq \epsilon
\end{aligned}
$$

where $\$(\cdot)$ is an oracle which outputs a random string of length $|\mathsf{enc}_{k'}(\cdot)|$.
That is, for a single encryption query, if the key is randomly picked, the output is indistinguishable from a random one.
The proof that our assumption is enough is a simple adaptation of the previous one.

D.3 The RUPAE security

After having proved the AE security in the blackbox model, we want to study the security when unverified plaintexts are released. We prove in this section that CONCRETE is RUPAE secure.

First we observe that, for CONCRETE, being CIML2 secure implies being INT-RUP secure, because, from the leakage the adversaries receives in decryption, that is, k_0^i, they are able to recompute the plaintext decrypted.

To prove the AE security, since we have proved that CONCRETE is AE secure and CIML2 (thus, INT-RUP) secure, it is enough to observe that during the decryption, a random ephemeral key k_0 is used. Thus, if using a random key, dec outputs a random result, we should be able to prove the RUPAE.

The separated syntax for CONCRETE First, we introduce the separated syntax for CONCRETE, that is, we have to define SDec and SVer: $\mathsf{SDec}_k(\cdot)$ is Dec_k without the three lines labeled as Verification in Fig. 4. That is, $\tilde{c}_0 = \mathsf{E}_{k_0}(p_B)$ is not

computed and there is not the check $\tilde{c}_0 \stackrel{?}{=} c_0$.
Instead $\mathsf{SVer}_k(\cdot)$ consists in the six lines labeled either as "common" or "verification" in Fig. 4. That is, $k_0 = \mathsf{F}^{*,-1}(h, c_{l+1})$ is recomputed. From it, $\tilde{c_0} = \mathsf{E}_{k_0}(p_B)$ is recomputed and $\tilde{c_0}$ is compared to c_0. If it is the case, $\mathsf{SVer}_k(\cdot)$ returns $\top$, otherwise, $\bot$.

Now we can prove the following

Theorem 8. *Let* F^* *be a* $(Q, t+Q(t_{\mathsf{H}}+(2L+1)t_{\mathsf{E}}), \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudorandom permutation (*STPRP*), let* E *be a* $(2, max(t', t''), \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* $(0, t + (Q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{E}}) + t_{\mathsf{f}(Q+1)}, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Then, the scheme* CONCRETE*, which encrypt messages which are at most* L*-block long, is* (q_E, q_D, t, ϵ)*-*RUPAE *secure with* ϵ *bounded by*

$$\epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + Q(L+1)\epsilon_{\mathsf{PRF}} + \frac{q_D}{2^n} + \frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}} + \frac{q_E(q_E-1)}{2^{n+1}}$$

where $Q = q_E + q_D$*,* t_{H} *is the time necessary to evaluate the hash function* H*,* t_{E} *is the time to compute* $y = \mathsf{E}_k(x)$*,* $t_{\mathsf{f}(Q)}$ *is the time needed to lazy sample a tweakable random permutation at most* Q *times and* t' *is bounded by*

$$t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} + 2(L+1)\max_{i=1,\ldots,q_D}[(i-1)t_{\$} + (q_D - i)t_{\mathsf{E}}] + \max_{j=0,\ldots,L+1}[(2J - 1 + \chi_0(J))t_{\$} + [2(L-J)-1]t_{\mathsf{E}}] + 2(L+1)q_E t_{\mathsf{E}}$$

and t'' *is bounded by*

$$t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} + q_D(2L+1)t_{\$} + 2\max_{j=0,\ldots,L}[jt_{\$} + (L-j)t_{\mathsf{E}}] + \max_{i=1,\ldots,q_E}[(i-1)(2L+1)t_{\$} + (q_E - i)(2L+1)t_{\mathsf{E}}]$$

and $\chi_0(J)$ *is the characteristic function of the set* $\{0\}$*.*

Following the same approach as in Sec. 5.2 we have that

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{RUPAE}}_{\Pi,\mathsf{A}} := \\
&\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$_E(\cdot),\$_D(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| = \\
&\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot),\perp(\cdot)} \Rightarrow 1\right] +\right. \\
&\quad \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\$_D(\cdot),\perp(\cdot)} \Rightarrow 1\right] + \\
&\quad \left.\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\$_D(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$_E(\cdot),\$_D(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| \leq \\
&\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| + \\
&\quad \left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\$_D(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| + \\
&\quad \left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\$_D(\cdot),\perp(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$_E(\cdot),\$_D(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| \leq \\
&\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| + \\
&\quad \left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\$_D(\cdot)} \Rightarrow 1\right]\right| + \\
&\quad \left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$_E(\cdot)} \Rightarrow 1\right]\right|
\end{aligned}
$$

where the last inequality is given by the fact that the oracle $\perp(\cdot)$ is easily emulated by anyone, while the oracle $\$(\cdot)$ is easily emulated picking random at most $q_D(L+2)n$ random bits (this costs time at most $(q_D(L+2))t_{\$}$ and no oracle queries). We use, similar to Thm. 2, $\overline{\Pi} = (\mathcal{K}, \overline{\mathsf{Enc}}, \overline{\mathsf{SDec}}, \overline{\mathsf{SVer}})$ which is the scheme $\overline{\Pi}$ used in that theorem, written with the separated syntax (in particular, for scheme $\overline{\Pi}$ we replace $\mathsf{F}^*(\cdot,\cdot)$ with $\mathsf{f}^*(\cdot,\cdot)$ and we suppose that there are no collisions for the hash function).
Now we have to separately study each piece. The first term

$$\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{SDec}_k(\cdot),\mathsf{SVer}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| = (\triangle)$$

can be easily bounded by the $\mathsf{CIML2}$ advantage we have already computed (Thm. 1). We use the same argument used in Thm. 2 to bound it. Thus, we bound it

$$(\triangle) \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{(q_D+1)(L+1)(Q+q_E)}{2^{n+1}} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}}$$

The second term has not been already bounded and it is the core of the proof.

$$\left|\Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\overline{\mathsf{Enc}}_{\mathsf{f}^*}(\cdot),\$_D(\cdot)} \Rightarrow 1\right]\right|$$

The idea of the proof is to prove that $\overline{\mathsf{SDec}}$ outputs a random string if the first ephemeral key k_0^i is randomly picked (This assumption is correct happens

because $k_0^i = \mathsf{f}^{*,-1}(\cdot,\cdot)$ where $\mathsf{f}^*(\cdot,\cdot)$ is a strong tweakable random permutation which is called always with different inputs [H is collision resistant and we have supposed that there are no collision for the hash function]).
Regarding the third term

$$\left|\Pr\left[\mathsf{A}^{\mathsf{Enc}^{\$}_{\mathsf{f}^*}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$^{\$}_{E}(\cdot)} \Rightarrow 1\right]\right|,$$

the proof of Thm 2 bounds it.

Although if we follow this pattern, we will be able to do a correct proof, we do few modifications to it in order to have a better bounds. In detail:

Proof. We use a sequence of games. As in other proofs, we use two progressive numeration for queries, one for encryption queries, the other for decryption queries.

Game 0 It is the Game where the adversary A has access to the oracles $\mathsf{Enc}^{\$}_k(\cdot), \mathsf{SDec}_k(\cdot)$ and $\mathsf{SVer}_k(\cdot)$. Let E_0 be the event that adversary A outputs 1 at the end of this game.

Game 1 It is Game 1 where we replace the STPRP $\mathsf{F}^*_k(\cdot)$ with the random tweakable permutation $\mathsf{f}^*(\cdot)$ and we suppose that all hash values h^i are different, provided that their inputs are different. Let E_1 be the event that A outputs 1 at the end of this game.

Bounding $|\Pr[E_0] - \Pr[E_1]|$ Using the proof of Thm. 1 we can easily bound

$$|\Pr[E_0] - \Pr[E_1]| \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}}.$$

Game 2 It is Game 1 where we replace the oracle $\overline{\mathsf{SDec}}_{\mathsf{f}^*}(\cdot)$ with $\$_D(\cdot)$ (A has still access to the real $\mathsf{SVer}_k(\cdot)$). Moreover, we suppose that also the blocks $\tilde{c_0}$ are random. Let E_2 be the event that A outputs 1 at the end of this game.

Transition between Game 1 and Game 2 We do it using $q_D(L+2)$ Games $1^{i,j}$ with $i = 1, ..., q_D$ and $j = -1, ..., L$.

Game $1^{I,J}$ It is Game 2 where for the first $I-1$ decryption queries, we have replaced all the blocks $(m_1^i, ..., m_{l^i}^i)$ and $\tilde{c_0}^i$ with random blocks. For the Ith decryption query, if $J = -1$, the plaintext m^I and $\tilde{c_0}^I$ are computed as usual; if $J = 0$ $\tilde{c_0}^I$ is random, while m^I is computed normally; otherwise, we have replaced the first $J-1$ blocks $(m_1^I, ..., m_J^I)$ and $\tilde{c_0}^I$ with random values. Moreover, at most $(I-1)(L+1) + J + 1$ random keys $k_0^i, ..., k_{l^i}^i$ for $i = 1, ..., I-1$ and k_j^I, $j = 0, ..., J$ are picked uniformly at random; in addition if $J \leq l^I$, a random key k_{J+1}^I is picked uniformly at random. Let $E_1^{I,J}$ be the event that adversary A outputs 1 at the end of this game.
We observe that for all $I = 1, ..., q_D$ Game $1^{I,0}$ and Game $1^{I,1}$ are the same in their outputs (thus, they are indistinguishable), but we need these games in

order to replace correctly the ephemeral key stream.
Moreover, we observe that Game 1 is Game $1^{1,0}$ while Game 2 is Game $1^{q_D,L+1}$.

Transition between Game $2^{I,J-1}$ and Game $2^{I,J}$ To do it we build a $(2,t')$-PRF adversary $\mathsf{B}^{I,J}$ against the PRF $\mathsf{E}(\cdot)$, with t' bounded by

$$t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} + 2(L+1)\max_{i=1,\ldots,q_D}[(i-1)t_{\$} + (q_D - i)t_{\mathsf{E}}] + \max_{j=0,\ldots,L+1}[(2J - 1 + \chi_0(J))t_{\$} + [2(L-J)-1]t_{\mathsf{E}}] + 2(L+1)q_E t_{\mathsf{E}}$$

where $t_{\mathsf{f}(Q)}$ is the time needed to lazy sample f^* Q times, t_{H} is the time needed to evaluate the hash function, $t_{\$}$ is the time needed to obtain a random block and t_{E} is the time needed to evaluate the PRF E.

The $(2,t')$-PRF adversary $\mathsf{B}^{I,J}$ adversary The PRF adversary has access to an oracle which is implemented either with $\mathsf{E}_{k_J^I}(\cdot)$ or with a random function $\mathsf{f}_J^I(\cdot)$. He has to distinguish the two situations. In detail:
First B_J^I picks 2 constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function $\mathsf{H} : \{0,1\}^* \longmapsto \{0,1\}^n$ and sends to A (p_A, p_B, H). He also picks a tweakable random permutation $\mathsf{f}^*(\cdot,\cdot)$ which he lazy samples.
When A does an encryption query on input m^i, with $m^i = (m_1^i, ..., m_{l^i}^i)$, the adversary B_J^I easily simulates it. In detail: B_J^I simply (1) picks uniformly at random r^i and sets $k_0^i := r^i$, then, (2) from the ephemeral key k_0^i, he computes $(c_0^i, ..., c_{l^i}^i)$, after that, (3) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i}^i)$ and finally (4) he computes $c_{l^i+1}^i = \mathsf{f}^*(h^i, k_0^i)$ and (5) $\mathsf{B}^{I,J}$ answers A c^i with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$. For every encryption queries $\mathsf{B}^{I,J}$ lazy samples f^* one time, moreover, he evaluates $\mathsf{E}(\cdot)$ $2(l^I+1) \leq 2(L+1)$ times, and he computes once the hash function H; thus, answering to A takes at most $t_{\mathsf{f}} + t_{\mathsf{H}} + 2(l^i+1)t_{\mathsf{E}}$ $[\leq t_{\mathsf{f}} + t_{\mathsf{H}} + 2(L+1)t_{\mathsf{E}}]$.
When A does a decryption query on input c^i, with $c^i = (c_0^i, c_1^i, ..., c_{l^i}^i, c_{l^i+1}^i)$, if $i < I$ $\mathsf{B}^{I,J}$ simply (1) computes the hash $h^i = \mathsf{H}(c_0^i, ..., c_l^i)$ and $k_0^i = \mathsf{f}^{*,-1}(h^i, c_{l^i+1}^i)$, (2) picks uniformly at random l^i blocks $(m_1^i, ..., m_{l^i}^i)$ with $|m_{l^i}^i| = |c_{l^i}^i|$, and answers m^i to A with $m^i = (m_1^i, ..., m_{l^i}^i)$. Moreover (3) he picks $l^i \leq L$ ephemeral keys , $k_1^i ..., k_{l^i}^i$ uniformly at random and a random block $\tilde{c}_0^i$. At the end, if $\tilde{c}_0^i = c_0^i$ he answers $\top$ to the following verification query, otherwise $\bot$.
For every of the first $I-1$ decryption queries $\mathsf{B}^{I,J}$ computes once the hash function H and lazy sample once f^*, he, also, picks uniformly at random $l^i+1 \leq L+1$ blocks and $l^i + 1 \leq L+1$ ephemeral keys; thus, answering to A takes at most $t_{\mathsf{H}} + t_{\mathsf{f}} + 2Lt_{\$}$ time.
When A does a decryption query on input c^i, with $c^i = (c_0^i, c_1^i, ..., c_{l^i}^i, c_{l^i+1}^i)$, if $i = I$, $\mathsf{B}^{I,J}$ simply (1) computes $h^I = \mathsf{H}(c_0^I \| ... \| c_{l^I}^I)$, if $J \neq 0$, he computes $k_0^I = \mathsf{f}^{*,-1}(h^I, c_{l^I+1}^I)$, picks at random $J-1$ blocks $(m_1^I, ..., m_{J-1}^I)$ with $|m_{l^I}^I| = |c_{l^I}^I|$ (if $J = 0$ he skips this step), then, (2) he queries his oracle on input p_A and p_B obtaining respectively k_{J+1}^I and y_J^I (if $J = L^I + 1$, $\mathsf{B}^{I,J}$ does not query his oracle on input p_A, but only on input p_B) after that, (3) he computes $m_J^I = y_J^I \oplus c_J^I$ (instead if $J = 0$ he sets $\tilde{c}_0^I = y_0^I$) then, (4) from k_{J+1}^I he com-

putes correctly $(m^I_{J+1}, ..., m^I_{l^I})$ and answers m^I to A with $m^I = (m^I_1, ..., m^I_{l^I})$. Moreover, (5) he picks $J-2$ ephemeral keys $k^I_1 ... k^I_{J-1}$ and a random block $\tilde{c}^i_0$ (if $J \neq 0$) uniformly at random. At the end, if $\tilde{c}^i_0 = c^i_0$ he answers $\top$ to the following verification query, otherwise $\bot$.
For the Ith decryption query, $\mathsf{B}^{I,J}$ queries at most twice his oracle (only once if $j = l^I$; otherwise twice), computes at most once the hash function H and lazy sample at most once f^*, he picks uniformly at random J blocks, one block is obtained via the oracle queries and he evaluates $2(l^I - J) \leq 2(L - J)$ times the PRF E. Moreover, he picks uniformly at random $max(J-1, 0)$ ephemeral keys; thus, answering to A takes at most $t_{\mathsf{H}} + t_{\mathsf{f}} + (2J - 1 + \chi_0(J))t_{\$} + [2(L^I - J) - 1]t_{\mathsf{E}}$ time ($\chi_0(J)$ is because if $J = 0$ no random value is picked and not -1 random value).
When A does a decryption query on input c^i, with $c^i = (c^i_0, c^i_1, ..., c^i_{l^i}, c^i_{l^i+1})$, if $i > I$, $\mathsf{B}^{I,J}$ simply (1) computes $h^i = \mathsf{H}(c^i_0, ..., c^i_{l^i}, c^i_{l^i+1})$, then (2) he computes $k^i_0 = \mathsf{f}^{*,-1}(h^i, c^i_{l^i+1})$ from which he (3) is able to correctly compute $m^i = (m^i_1, ..., m^i_{l^i})$ and $\tilde{c}^i_0$. He answers m^i to A. At the end, if $\tilde{c}^i_0 = c^i_0$ he answers $\top$ to the following verification query, otherwise $\bot$.
For every of the last $q_D - I$ decryption queries, $\mathsf{B}^{I,J}$ evaluates one time the hash function $\mathsf{H}(\cdot)$, he lazy samples once $\mathsf{f}^*(\cdot, \cdot)$ and evaluates the PRF E $2l^i + 1 \leq 2L + 1$ times; thus, answering to A takes at most $t_{\mathsf{f}} + t_{\mathsf{H}} + [2L + 1]t_{\mathsf{E}}$ time.
When A outputs a bit, $\mathsf{B}^{I,J}$ outputs the same bit.
Thus in total the adversary $\mathsf{B}^{I,J}$ does 2 queries to his oracle, moreover he runs in time bounded by

$$t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} - \{2[(I-1)(L+1) + J] - 1 + \chi_0(J)\}t_{\$} + 2[(L-J) + (Q-I)(L+1)]t_{\mathsf{E}} \leq$$
$$t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} + 2(L+1) \max_{i=1,...,q_D} [(i-1)t_{\$} + (q_D - i)t_{\mathsf{E}}] +$$
$$\max_{j=0,...,L+1} [(2J - 1 + \chi_0(J))t_{\$} + [2(L-J) - 1]t_{\mathsf{E}}] + 2(L+1)q_E t_{\mathsf{E}}$$

Ephemeral key collisions We observe that, if the key k^I_J used in the oracle is the same as one of the ephemeral keys picked randomly, or an ephemeral key used in an encryption query, there is a problem, because, in such a case the outputs of the oracle are not consistent. Thus, we define the event of ephemeral key collision (EKC)

$$EKC^{I,J} := \left\{ \begin{array}{c} k^I_J = k^i_j \text{ for an } i = 1, ..., I-1 \\ \text{or if } i = I \text{ and } j = 0, ..., J-1 \\ \text{or if } i \text{ belongs to an encryption query} \end{array} \right\}$$

If the event $EKC^{I,J}$ happens in Game $1^{I,J-1}$, it creates a problem to simulate correctly Game $1^{I,J}$. In Game $1^{I,J}$ since k^i_J is picked uniformly at random $\Pr[EKC^{I,J}] \leq \frac{(L+1)(I-1+q_E)+J}{2^n}$.

Bounding $|\Pr[E^{I,J-1}_1] - \Pr[E^{I,J}_1]|$ After having supposed that event $EKC^{I,J}$ does not happens, we observe that $\mathsf{B}^{I,J}$ simulates Game $1^{I,J-1}$ if the oracle is

implemented with $\mathsf{E}_{k_j^I}(\cdot)$ otherwise Game $1^{I,J}$. Since $\mathsf{E}(\cdot)$ is a $(2, t', \epsilon_{\mathsf{PRF}})$-PRF we can bound

$$|\Pr[E_1^{I,J-1}] - \Pr[E_1^{I,J}]| \leq \epsilon_{\mathsf{PRF}} + \Pr[EKC^{I,J}] \leq$$

$$\epsilon_{\mathsf{PRF}} + \frac{(L+1)(I-1+q_E)+J}{2^n}$$

Transition between Game $1^{I,L+1}$ and Game $1^{I+1,0}$ As in the proof of Thm. 2, Game $1^{I,L+1}$ and Game $1^{I+1,0}$ are the same, since we have replaced with random the same plaintext blocks and the others are computed in the same way. Thus, $\Pr[E_1^{I,L+1}] = \Pr[E_1^{I+1,0}]$.

Bounding $|\Pr[E_1] - \Pr[E_2]|$ Using the previous 2 transitions and iterating them, we are finally able to bound $|\Pr[E_1] - \Pr[E_2]|$

$$\begin{aligned}
&|\Pr[E_1] - \Pr[E_2]| \leq \\
&\sum_{i=1}^{q_D}\left(\sum_{j=0}^{L}|\Pr[E_1^{i,j-1}] - \Pr[E_1^{i,j}]|\right) + \sum_{i=1}^{q_D-1}|\Pr[E_1^{i,L+1}] - \Pr[E_1^{i+1,0}]| \leq \\
&\qquad \sum_{i=1}^{q_D}\left(\sum_{j=0}^{L}\epsilon_{\mathsf{PRF}} + \frac{(L+1)(i-1+q_E)+j}{2^n}\right) + 0 \leq \\
&q_D(L+1)\epsilon_{\mathsf{PRF}} + \frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}} - \frac{(L+1)q_E[(L+1)q_E-1]}{2^n}
\end{aligned}$$

where the last inequality is given by the fact that

$$\sum_{i=1}^{q_D}\sum_{j=0}^{L}[(L+1)(i-1+q_E)+j] = \sum_{\lambda=(L+1)q_E}^{(L+1)Q-1}\lambda = \sum_{\lambda=0}^{(L+1)Q-1}\lambda - \sum_{\lambda=0}^{(L+1)q_E-1}\lambda$$

and, by the well known fact, that the sum of the first M natural numbers is

$$\sum_{i=1}^{M} i = \frac{M(M+1)}{2}$$

and $Q = q_E + q_D$.

Game 3 It is Game 2 where we replace the $\overline{\mathsf{SVer}}_{\mathsf{f}^*}(\cdot)$ oracle with the $\perp(\cdot)$ oracle. Let E_3 be the event that A outputs 1 at the end of this game.

Transition between Game 2 and Game 3. We build $q_D + 1$ games Game 2^i, for $i = 0, ..., q_D$.

Game 2^i It is Game 2 where the $\overline{\mathsf{SVer}}_{\mathsf{f}^*}(\cdot)$ oracle answers $\perp$ to the first i decryption queries, without evaluating them. We observe that Game 2^0 is Game

2, while Game 2^{q_D} is Game 3. Let E_2^i be the event that A outputs 1 at the end of Game 2^i.

Transition between Game 2^i and Game 2^{i+1} Consider the following event (valid ciphertext):

$$VC^i := \{ \text{ ciphertext } c^i \text{ is valid}\} = \{\tilde{c}_0^i = c_0^i\}.$$

Clearly if event VC^i does not happen, Game 2^{i-1} and Game 2^i are identical. Moreover, since in Game 2 all the $\tilde{c}_0^i$ are picked uniformly at random $\Pr[VC^i] = \frac{1}{2^n}$. Thus, we can bound $\left|\Pr[E_2^{i-1}] - \Pr[E_2^i]\right|$.

Bounding $\left|\Pr[E_2^{i-1}] - \Pr[E_2^i]\right|$

$$\left|\Pr[E_2^{i-1}] - \Pr[E_2^i]\right| = \Pr[VC^i] = \frac{1}{2^n}$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$ Adding the previous bounds $\left|\Pr[E_2^{i-1}] - \Pr[E_2^i]\right|$ we obtain:

$$|\Pr[E_2] - \Pr[E_3]| = \sum_{i=0}^{q_D} \left|\Pr[E_2^{i-1}] - \Pr[E_2^i]\right| = \sum_{i=1}^{q_D} \frac{1}{2^n} = \frac{q_D}{2^n}$$

Game 4 It is Game 3 where we replace the oracle $\overline{\mathsf{Enc}}_{f^*}(\cdot)$ with $\$_E(\cdot)$. Let E_4 be the event that A outputs 1 at the end of this game.

Transition between Game 3 and Game 4 It is similar to the transition between Game 1 and 2. We do it using $q_E(L+2)$ games Games $3^{i,j}$ with $i = 1, ..., q_E$ and $j = -1, ..., L+1$.

Game $3^{I,J}$ It is Game 3 where for the first $I-1$ encryption query, we have replaced all the blocks $(c_0^i, ..., c_{l^i}^i)$ with random blocks (we do not touch the last block $c_{l^i}^i$). For the Ith encryption query we have replaced the first J blocks $(c_0^I, ..., c_J^I)$ with random values. Moreover, at most $(I-1)(L+1)+J+1$ random keys $k_0^i, ..., k_{l^i}^i$ for $i = 1, ..., I-1$ and k_j^I, $j = 0, ..., J$ are picked uniformly at random; in addition if $J \leq l^I$, a random key k_{J+1}^I is picked uniformly at random. Let $E_3^{I,J}$ be the event that adversary A outputs 1 at the end of this game.
We observe that Game 3 is Game $3^{1,-1}$ while Game 4 is Game $3^{q_E,L+1}$.

Transition between Game $3^{I,J-1}$ and Game $3^{I,J}$ for $J < L+2$ To do it, for $J < L+2$, we build a $(2, t'')$-PRF adversary $\mathsf{C}^{I,J}$ against the PRF $\mathsf{E}(\cdot)$, with $t'' \leq$

$$t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} + q_D(2L+1)t_{\$} + 2\max_{j=0,...,L}[jt_{\$} + (L-j)t_{\mathsf{E}}] + \max_{i=1,...,q_E}[(i-1)(2L+1)t_{\$} + (q_E - i)(2L+1)t_{\mathsf{E}}]$$

The $(2, t'')$**-PRF adversary** $\mathsf{C}^{I,J}$ **adversary** The PRF adversary has access to an oracle which is implemented either with $\mathsf{E}_{k_J^I}(\cdot)$ or with a random function $\mathsf{f}_J^I(\cdot)$. He has to distinguish the two situations. In detail:
First, $\mathsf{C}^{I,J}$ picks 2 constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function $\mathsf{H} : \{0,1\}^* \longmapsto \{0,1\}^n$ and sends to A (p_A, p_B, H). He also picks a tweakable random permutation $\mathsf{f}^*(\cdot,\cdot)$ which he lazy samples.
When A does an encryption query on input m^i, with $m^i = (m_1^i, ..., m_{l^i}^i)$, if $i < I$ the adversary C_I^J proceeds as follow: first, (1) he picks uniformly at random r^i and he sets $k_0^i := r^i$, then, (2) he picks uniformly at random $l^i + 1$ blocks $(c_0^i, c_1^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ with $|c_{l^i}^i| = |m_{l^i}^i|$, and (5) $\mathsf{C}^{I,J}$ answers c^i to A with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$. Finally, (6) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i}^i)$ and lazy samples $c_{l^i+1}^i = \mathsf{f}^*(h^i, k_0^i)$. Moreover, (7) he picks l^i random keys $k_1^i, ..., k_{l^i}^i$ uniformly at random (we observe that also k_0^i, being equal to the randomness r^i is picked uniformly at random). For the first $I-1$ encryption queries $\mathsf{C}^{I,J}$ computes one the hash function H, lazy samples once f^*, picks at random $l^i+1 < L+1$ random blocks and $l^i + 1 < L + 1$ ephemeral keys; thus, answering to A takes at most $t_{\mathsf{H}} + t_{\mathsf{f}^*} + 2(l^i + 1)t_{\$}$ $[\leq t_{\mathsf{H}} + t_{\mathsf{f}^*} + 2(L + 1)t_{\$}]$ time.
When A does the Ith encryption query on input $m^I = (m_1^I, ..., m_{l^I}^I)$, C_J^I proceeds as follow: first, (1) he picks uniformly at random r^i and he $k_0^i := r^i$, then, (2) he picks uniformly at random J blocks $c_0^I, ..., c_{J-1}^I$, after that (3) he calls his oracle on inputs p_A and p_B receiving respectively k_{J+1}^I and y_J^I (if $J = L^I$, C_J^I queries his oracle only on input p_B) then, (4) he computes $c_J^I = y_J^I \oplus m_J^I$ (if $J = 0$, $c_0^I = y_0^I$), after that, (5) from k_{J+1}^I, C_J^I is able to correctly compute $c_{J+1}^I, ..., c_{l^I}^I$, then (6), if $J < l^I + 1$, he computes the hash $h^I = \mathsf{H}(c_0^I \| ... \| c_{l^I}^I)$ and $c_{l^I+1}^I = \mathsf{f}^*(h^I, r^I)$, otherwise he picks a random block $c_{l^I+1}^I$, finally, (7) he answers $c^I = (c_0^I, ..., c_{l^I+1}^I)$ to A. Moreover, (8) he picks J random keys $k_0^I, k_1^I, ..., k_{J-1}^I$ uniformly at random (k_0^I is picked uniformly at random, being equal to r^I). When A does the Ith encryption query, C_J^I lazy samples $\mathsf{f}^*(\cdot,\cdot)$ one time, computes once the hash function H, moreover, he picks at random J random blocks and $max(J, 1)$ random keys, he evaluates the PRF E $2(l^I - J) \leq 2(L - J)$ times, thus answering to A takes time bounded by $t_{\mathsf{f}} + t_{\mathsf{H}} + 2(L - J)t_{\mathsf{E}} 2Jt_{\$}$.
When A does an encryption query on input m^i, with $m^i = (m_1^i, ..., m_{l^i}^i)$, if $i > I$ the adversary C_I^J easily simulates it. In detail: C_J^I simply (1) picks uniformly at random r^i and sets $k_0^i := r^i$, then, (2) from the ephemeral key k_0^i, he computes $(c_0^i, ..., c_{l^i}^i)$, after that, (3) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i}^i)$ and finally (4) he computes $c_{l^i+1}^i = \mathsf{f}^*(h^i, k_0^i)$ and (5) $\mathsf{C}^{I,J}$ answers c^i to A with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$. For every of the last $q_E - I$ encryption queries $\mathsf{C}^{I,J}$ lazy samples f^* one time, moreover, he evaluates $\mathsf{E}(\cdot)$ $2(l^I + 1) \leq 2(L + 1)$ times, and he computes once the hash function H; thus, answering to A takes at most $t_{\mathsf{f}} + t_{\mathsf{H}} + 2(l^i + 1)t_{\mathsf{E}}$ $[\leq t_{\mathsf{f}} + t_{\mathsf{H}} + 2(L + 1)t_{\mathsf{E}}]$ time, with t_{f} the time necessary to lazy sample $\mathsf{f}^*(\cdot,\cdot)$.
When A does a decryption query on input c^i, with $c^i = (c_0^i, c_1^i, ..., c_{l^i}^i, c_{l^i+1}^i)$, $\mathsf{C}^{I,J}$ simply (1) picks uniformly at random l^i blocks $(m_1^i, ..., m_{l^i}^i)$ with $|m_{l^i}^i| = |c_{l^i}^i|$, and answers A (m^i) with $m^i = (m_1^i, ..., m_{l^i}^i)$. Moreover, (2) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i}^i)$ and lazy samples $k_0^i = \mathsf{f}^{*,-1}(h^i, c_{l^i+1}^i)$. Finally, (3) he picks

$l^i \leq L$ ephemeral keys $k_1^i ..., k_{l^i}^i$ uniformly at random. Then, he answers $\perp$ to the following verification query.
For every decryption queries $\mathsf{C}^{I,J}$ lazy samples $\mathsf{f}^*(\cdot,\cdot)$ one time, computes once the hash function H, picks uniformly at random $l^i \leq L$ blocks and $l^i \leq L$ ephemeral keys; thus, answering to A takes at most $t_{\mathsf{H}} + t_{\mathsf{f}} + (2L+1)t_{\$}$ time.
When A outputs a bit, $\mathsf{C}^{I,J}$ outputs the same bit.
Thus in total the adversary $\mathsf{C}^{I,J}$ does 2 queries to his oracle, moreover he runs in time bounded by

$$\begin{aligned} t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} + [q_D(2L+1) + (I-1)(2+1) + 2J]t_{\$} + \\ 2[(L-J) + (q_E - I)(L+1)]t_{\mathsf{E}} \leq \\ t + t_{\mathsf{f}(Q)} + Qt_{\mathsf{H}} + q_D(2L+1)t_{\$} + 2\max_{j=0,...,L}[jt_{\$} + (L-j)t_{\mathsf{E}}] + \\ \max_{i=1,...,q_E}[(i-1)(2L+1)t_{\$} + (q_E - i)(2L+1)t_{\mathsf{E}}] \end{aligned}$$

Ephemeral key collisions We observe that if the key k_J^I used in the oracle is the same as one of the ephemeral key picked randomly there is a problem, because the simulation would not be correct (we have already supposed that all epehemeral keys, used in decryption queries, were different). Thus, we define the event of ephemeral key collision (EKC)

$$EKC^{I,J} := \left\{ \begin{array}{c} k_J^I = k_j^i \text{ for an } i = 1, ..., I-1 \text{ and any } j \\ \text{or if } i = I \text{ and } j = 0, ..., J-1 \end{array} \right\}$$

We observe that event EKC covers also the event of a collision in the randomnesses (being them equal to k_0^i).
If the event $EKC^{I,J}$ happens in Game $3^{I,J'1}$, it would be problematic to simulate correctly Game $3^{I,J}$. In Game $3^{I,J}$ since k_J^I is picked uniformly at random $\Pr[EKC^{I,J}] \leq \frac{(L+1)(I-1)+J+1}{2^n}$.

Bounding $|\Pr[E_3^{I,J-1}] - \Pr[E_3^{I,J}]|$ for $J \leq L+1$ After having supposed that event $EKC^{I,J}$ does not happens, we observe that $\mathsf{C}^{I,J}$ simulates Game $3^{I,J-1}$ if the oracle is implemented with $\mathsf{E}_{k_J^I}(\cdot)$ otherwise Game $3^{I,J}$. Since $\mathsf{E}(\cdot)$ is a $(2, t'', \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ we can bound

$$|\Pr[E_3^{I,J-1}] - \Pr[E_3^{I,J'1}]| \leq \epsilon_{\mathsf{PRF}} + \Pr[EKC^{I,J}] \leq$$

$$\epsilon_{\mathsf{PRF}} + \frac{(L+1)(I-1)+J}{2^n}$$

Transition between Game $3^{I,L}$ and Game $3^{I,L+1}$ The difference between Game $3^{I,L}$ and Game $3^{I,L}$ is how the block $c_{l^I+1}^I$ is computed. But since $c_{l^I+1}^I$ is in both games randomly picked, but in the first according to a tweakable random permutation with fresh inputs, thus, using the lemma about switching a PRP with a PRF $|\Pr[E_3^{I,L}] - \Pr[E_3^{I,L+1}]| = \frac{q-D+I-1}{2^n}$.

Transition between Game $3^{I,L+1}$ and Game $3^{I+1,-1}$ As in the proof of Thm. 2, Game $3^{I,L+1}$ and Game $3^{I+1,-1}$ are the same, since we have replaced with random the same ciphertext blocks and the others are computed in the same way. Thus, $\Pr[E_3^{I,L+2}] = \Pr[E_3^{I+1,-1}]$.

Bounding $|\Pr[E_3]-\Pr[E_4]|$ Using the previous 3 transitions and iterating them, we are finally able to bound $|\Pr[E_3] - \Pr[E_4]|$

$$
\begin{aligned}
|\Pr[E_3] - \Pr[E_4]| \leq \sum_{i=1}^{q_E}\left(\sum_{j=0}^{L}|\Pr[E_3^{i,j-1}] - \Pr[E_3^{i,j}]|\right) + \\
\sum_{i=1}^{q_E}|\Pr[E_3^{i,L}] - \Pr[E_3^{i,L+1}]| + \sum_{i=1}^{q_E-1}|\Pr[E_3^{i,L+1}] - \Pr[E_3^{i+1,-1}]| \leq \\
\sum_{i=1}^{q_E}\left(\sum_{j=0}^{L}\epsilon_{\mathsf{PRF}} + \frac{(L+1)(q_D+i-1)+j}{2^n}\right) + \frac{I-1}{2^n} + 0 \leq \\
q_E(L+1)\epsilon_{\mathsf{PRF}} + \sum_{i=1}^{q_E}\left(\sum_{j=1}^{L+1}\frac{(L+1)(i-1)+j}{2^n} + \frac{q_D+I-1}{2^n}\right) \leq \\
q_E(L+1)\epsilon_{\mathsf{PRF}} + \sum_{\lambda=0}^{(L+1)q_E-1}\frac{\lambda}{2^n} + \frac{q_Dq_E}{2^n} + \sum_{i=0}^{q_E-1}\frac{i}{2^n} \leq \\
q_E(L+1)\epsilon_{\mathsf{PRF}} + \frac{(L+1)q_E[(L+1)q_E-1]}{2^{n+1}} + \frac{q_Dq_E}{2^n} + \frac{q_E[q_E-1]}{2^{n+1}}
\end{aligned}
$$

where the second to last inequality is obtained observing that

$$
\sum_{i=1}^{q_E}\left(\sum_{j=0}^{L}L(i-1)\right) = \sum_{\lambda=0}^{(L+1)q_E-1}\lambda
$$

is the sum of the natural numbers in $[0,(L+1)q_E-1]$.

Bounding $\Pr[E_0]$ This concludes the proof since using all the bounds computed, we are finally able to bound $\Pr[E_0]$

$$
\begin{aligned}
\Pr[E_0] \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + q_D(L+1)\epsilon_{\mathsf{PRF}} + \frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}} - \\
\frac{(L+1)q_E[(L+1)q_E-1]}{2^n} + \frac{q_D}{2^n} + q_E(L+1)\epsilon_{\mathsf{PRF}} + \\
\frac{(L+1)q_E[(L+1)q_E-1]}{2^{n+1}} + \frac{q_Dq_E}{2^n} + \frac{q_E[q_E-1]}{2^{n+1}} \leq \\
\epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + Q(L+1)\epsilon_{\mathsf{PRF}} + \frac{q_D}{2^n} + \frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}} + \frac{q_E(q_E-1)}{2^{n+1}}
\end{aligned}
$$

Observation We do not see completely the term $\frac{Q(Q-1)}{2^{n+1}}$ needed to replace a PRP with a PRF, but this is due to the fact that part of it is in the term $\frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}}$, which forces all the ephemeral keys used to be different.

Modification to the previous pattern . We replaced before SDec_k than $\mathsf{SVer}_k(\cdot)$ because we can replace $\mathsf{E}_{k_0^i}(p_A)$ and $\mathsf{E}_{k_0^i}(p_B)$ at the same moment in every decryption query (so, we did not have to "recompute k_1^i in every decryption query in the previous proofs (Thm. 1 and Thm. 2)).

Security requirement for a generic rekeyed encryption scheme If, instead of PSV we had used another rekeyed encryption scheme $\Pi' = (\mathcal{K}, \mathsf{enc}', \mathsf{dec}')$, to have the RUPAE security, we need that Π' has the following property:
for every $(0, t)$-bounded adversary A the advantage

$$\mathsf{Adv}_{\Pi',\mathsf{A}} := \Big|\Pr[\mathsf{A}(m) \Rightarrow 1; k' \xleftarrow{\$} \mathcal{K}', m \leftarrow \mathsf{A}, m \leftarrow \mathsf{dec}_{k'}(m)] - \Pr[\mathsf{A}(m) \Rightarrow 1; c \leftarrow \mathsf{A}, m \leftarrow \$(c)]\Big| \leq \epsilon$$

where $\$(\cdot)$ is an oracle which outputs a random string of length $|\mathsf{dec}_{k'}(\cdot)|$.
That is, for a single decryption query, if the key is randomly picked, the output is indistinguishable from a random one.
The proof that this is enough is a simple adaptation of the previous one.

D.4 CPAL2 and CCAL2 security

Now we want to study the confidentiality with leakage. In this section, we prove that CONCRETE is CPAL2 and CCAL2 with the simulatable hypothesis for the leakage of E, reducing the CPAL2 and CCAL2 security of the whole scheme to the eavesdropper security with leakage of an ideal variant of PSV which encrypts a single block.
First, we study the source of leakage, then, we introduce the ideal single block variant of PSV (called PSVs^I) and we reduce the Eavesdropper security of PSVs, (that is, the single block variant of PSV) to that of PSVs^I. After that, we can prove the Eavesdropper security with leakage of the whole PSV scheme based on that of PSVs^I. Then, we study and bound the leakage of the first block and we are finally able to prove the eavesdropper security with leakage of the whole CONCRETE. From this, we are able to prove the CPAL2 and CCAL2 security of CONCRETE.

Leakage We have four components that can leak: the PRF $\mathsf{E}(\cdot)$ the $\oplus(\cdot,\cdot)$ used to obtain the ciphertexts, the STPRP leak free $\mathsf{F}^*(\cdot,\cdot)$ (the latter may only leak its inputs and outputs, not the key used) and the random generator used internally by CONCRETE. We observe that, even if there is an hash function, we do not consider its leakage, since its key and its inputs and outputs, as well, are known by the adversary $[h = \mathsf{H}(c_0\|...\|c_l)]$.
We suppose that E is implemented with a simulatable leakage.

The EavL2 security of scheme CONCRETE The aim of this section is to reduce the $\mathsf{EavL2s}$ security of the whole scheme to the security of an ideal encryption scheme which encrypts a single block. To do this, first, we study the security of PSV for a single block (PSVs) and then we use this result to prove the security for the whole PSV. Then, we study the security of the first block and we finish combining everything to obtain the given bound.
The proof is based on the original one for PSV [33]. We observe that given a message m, and $c \leftarrow \mathsf{Enc}_k(m)$, the ciphertext c can be seen as $\mathsf{PSV}_{k_0}(0^n \| m) \| \mathsf{F}^*_k(h, k_0)$, for an ephemeral key k_0 picked uniformly at random.

Single block PSV First we consider PSVs, defined in Tab. 7. This is a version of PSV where only one block is encrypted. With respect to the standard PSV scheme an additional output is given k_1, that is, the refreshed key. This is useful to compose PSVs in order to obtain PSV and does not compromise the security. Its leakage is given by the leakage given by the leakage of E, $\oplus$ and by the leakage of the generation of k_0, which we simulate via a simulator $\mathcal{S}^{\mathsf{L}}$ which should simulate the leakage of k_0 when is generated via $\mathsf{E}_{k^-}(p_A)$ using a random key k^-.
Let us consider the $\mathsf{EavL2s}$ experiment, defined in Tab. 8. We want to compute the probability that a $(0, t)$-adversary is able to win this experiment. To compute this, we introduce an idealized version of PSVs, PSVs^I. We say that a scheme is (t, ϵ)-$\mathsf{EavL2(s)}$-secure if for every (t)-adversary, the probability that he wins the $\mathsf{EavL2(s)}$ game is bounded by $\frac{1}{2} + \epsilon$.

Idealized version single block PSVs^I We idealize PSVs, giving birth to PSVs^I, which is described in the right column of Tab. 7. Instead of obtaining y and k_1 via the PRF E, they are picked at uniformly at random.
For the leakage, the leakage of E is replaced with the leakage produced by its simulator.
We consider again the probability that a $(0, t)$ adversary is able to win the $\mathsf{EavL2s}$ experiment (Tab. 8).

Relation between $\Pr[\mathsf{A}$ **wins** $\mathsf{EavL2s}_{\mathsf{PSVs}}]$ **and** $\Pr[\mathsf{A}$ **wins** $\mathsf{EavL2s}_{\mathsf{PSVs}^I}]$ We can prove that the real and ideal version of the single block version are indistinguishable with the following lemma:

Lemma 1. *Let* $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \longmapsto \{0,1\}^n$ *be a* $(2, t - q_{\mathsf{L}} t_{\mathsf{L(E)}} - 3t_{\mathcal{S}} - t_{\mathsf{L}(\oplus)} - 2t_{\$}, \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ *whose implementation has a leakage function* L *which has* $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_{\mathsf{L}}, t - q_{\mathsf{L}} t_{\mathsf{L(E)}} - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)} - t_{\$}, \epsilon_{\mathsf{2\text{-}sim}})$-*2-simulatable leakage and let* $\mathcal{S}^{\mathsf{L}}$ *be an appropriate* $(q_{\mathcal{S}}, t_{\mathcal{S}})$-*bounded leakage simulator. Then, for every* $m, p_A, p_B \in \{0,1\}^n$ *with* $p_A \neq p_B$*, and every* (q_{L}, t) *adversary* A^{L}*, the following holds:*

$$\left|\Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSVL}s_{k_0}(m)) \Rightarrow 1] - \Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSVL}s^I_{k_0}(m)) \Rightarrow 1]\right| \leq$$

$$\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}}$$

where t_{sim} *is the time needed to relay the content of the two* $\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}$ *and the* $\mathsf{Gen}\text{-}\mathcal{S}$ *query (that is, the queries needed in the simulatable game, see Tab. 5),*

PSVs and PSVsfb: the single block variant of PSV and the single block variant for the first block and their ideal counterparts PSVs^I (and PSVsfb^I)	
Gen: $k_0 \leftarrow \{0,1\}^n$ $[\ k \leftarrow \{0,1\}^n\ ]$ $\mathsf{encs}_{k_0,k}(m)$: $[\mathsf{encsfb}_{k_0,k}(h)$:] $y = \mathsf{E}_{k_0}(p_B)$ $c = y \oplus m$ $[c = y]$ $k_1 = \mathsf{E}_{k_0}(p_A)$ $[\ d = \mathsf{F}^*_k(h, k_0)]$ Return (c, k_1) $[(c, k_1, d)]$ $\mathsf{decs}_{k_0}(c)$ and $\mathsf{decsfb}_{k_0,k}$ proceed in the natural way.	Gen^I : $k_0 \leftarrow \{0,1\}^n$ $\mathsf{encs}^I(m)$: $[\mathsf{encsfb}^I_{k_0}(h)$:] $y \xleftarrow{\$} \{0,1\}^n$ $c = y \oplus m$ $[c = y]$ $k_1 \xleftarrow{\$} \{0,1\}^n$ $[\ d \leftarrow \{0,1\}^n]$ Return (c, k_1) $[(c, k_1, d)]$ $\mathsf{decs}^I_{k_0}(c)$ and $\mathsf{decsfb}^I_{k_0,k}$ proceed in the natural way.
The leakage resulting from $\mathsf{encs}_{k_0}(m)[\mathsf{encsfb}_{k_0,k}(m,h)]$ is defined as $\mathsf{L}_{\mathsf{encs}}(m;k_0) := (\mathsf{L}_E(p_A;k_0), \mathsf{L}_E(p_B;k_0), \mathsf{L}_\oplus(m,y), \mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-)$ $[\mathsf{L}_{\mathsf{encsfb}}(h;k_0,k) := (\mathsf{L}_E(p_A;k_0), \mathsf{L}_E(p_B;k_0), \mathsf{L}_\oplus(m,y), \mathsf{L}_\$(k_0), \mathsf{L}_{\mathsf{F}^*}(k_0,h;k))]$ with $k^- \xleftarrow{\$} \{0,1\}^n$.	The leakage resulting from $\mathsf{encs}^I(m)[\mathsf{encsfb}^I_k(m,h)]$ is defined as $\mathsf{L}_{\mathsf{encs}^I}(m,k_1,y;k_0) := (\mathcal{S}^{\mathsf{L}}(k_0,p_A,k_1), \mathcal{S}^{\mathsf{L}}(k_0,p_B,y), \mathsf{L}_\oplus(m,y), \mathcal{S}^{\mathsf{L}}(k^-,p_A,k_0), k^-)$ $[\mathsf{L}_{\mathsf{encsfb}^I}(h,k_1,y;k_0,k) := (\mathcal{S}^{\mathsf{L}}(k_0,p_A,k_1), \mathcal{S}^{\mathsf{L}}(k_0,p_B,k_1), \mathsf{L}_\oplus(m,y), \mathsf{L}_\$(k_0), \mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k,h,k_0,d))]$ with $k^- \xleftarrow{\$} \{0,1\}^n$.

Table 7. PSVs: the single block variant of PSV. The variant for the first block, PSVsfb^I, adds the element in the square brackets

$t_{\mathsf{L}(\mathsf{E})}$ is the time needed to collect the leakage of a call to E, $t_{\mathsf{L}(\oplus)}$ is the time needed to collect the leakage of XOR of two blocks, $t_{\mathcal{S}}$ is the time needed to simulate one leakage query and $t_\$$ is the time needed to collect a random block.

The proof is identical to the proof of Lemma 2 [33] which, is inspired by Lemma 1 [37].

Proof. Again we use a sequence of games:

Game 0 Let Game 0 be the game where the adversary $\mathsf{A}^{\mathsf{L}}(p_A, p_B)$ is going against scheme PSVs_{k_0} choosing the message m and receiving the ciphertext c, the rekeyed key k_1 and the leakage $\mathsf{L}_{\mathsf{encs}}(m;k_0)$ for a random key k_0. He is given the constants p_A, p_B. (That is, it is an eavesdropper game with leakage where the adversary outputs a single plaintext). The adversary A^{L} is granted to q_{L} queries to the leakage oracle.
Let E_0 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Game 1 Let Game 1 be Game 0 where we have replaced the leakages $(\mathsf{L}_E(p_A;k_0), \mathsf{L}_E(p_B;k_0), \mathcal{S}^{\mathsf{L}}(k^-,p_A,k_0))$ with $(\mathcal{S}^{\mathsf{L}}(k^*,p_A,k_1), \mathcal{S}^{\mathsf{L}}(k^*,p_B,y), \mathcal{S}^{\mathsf{L}}(k^-,p_A,k^*))$ for a random key k^* in the leakage of encs_{k_0}.

The $\mathsf{EavL2s}_{\mathsf{PSVs},\mathsf{L_E},\mathsf{L}_\oplus,\mathsf{A}}(1^n)$ and $\mathsf{EavL2s}_{\mathsf{PSV},\mathsf{L_E},\mathsf{L}_\oplus,\mathsf{A}}(1^n)$ experiments

Initialization: $k_0 \leftarrow \mathcal{K}$ s.t. $\lvert k\rvert = n$, $p_A, p_B \in \{0,1\}^n$ $b \xleftarrow{\$} \{0,1\}$	Oracle $\mathsf{encs}\mathsf{L}_{k_0}(m^*)$: $c \leftarrow \mathsf{encs}_{k_0}(m^*)$ $\mathsf{L}_{encs} = \mathsf{L}(\mathsf{encs}_{k_0}(m^*)$ Return (c, L_{encs})
Challenge output for PSVs $[for$ PSVs$]$: $(m^{*,0}, m^{*,1}) \leftarrow \mathsf{A}^{\mathsf{L}}(p_A, p_B)$ If $\lvert m^{*,0}\rvert \neq \lvert m^{*,1}\rvert$ Return 0 [If $\lvert m^{*,0}\rvert = \lvert m^{*,1}\rvert \neq n$ Return 0] Else $((c,k_1),\mathsf{L}) \leftarrow \mathsf{enc}[s]\mathsf{L}_{k_0}(m^{*,b})$; Finalization: $b' \leftarrow \mathsf{A}((c,k_1),\mathsf{L})$ If $b = b'$, Return 1 Return 0	Oracle $\mathsf{encL}_{k_0}(m^*)$: $c \leftarrow \mathsf{enc}_{k_0}(m^*)$ $\mathsf{L}_{encs} = \mathsf{L}(\mathsf{enc}_{k_0}(m^*)$ Return $(c, \mathsf{L}_{\mathsf{enc}})$

Table 8. The EavL2s experiment agaisnt scheme PSVs (adding the part in bracket) and against PSV.

Let E_1 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Transition between Game 0 and 1 We build a $(q_{\mathsf{L}}, t - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)} - t_{\$})$-adversary B^{L} against the 2-simulatability of E based on A to bound the absolute difference $|\Pr[E_0] - \Pr[E_1]|$.

The $(q_{\mathsf{L}}, t - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)} - t_{\$})$ 2-simulatability adversary B^{L} B^{L} is playing the 2-simulatability game (see Sec. **??** and Tab. 5) and he has to simulate either Game 0 or Game 1 for A^{L}. At the start of the game, B^{L} chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ which he gives to A^{L}.
When A^{L} does a leakage query, B^{L} does the same query and relays its answer to A^{L}.
When A^{L} asks his query on input m, B^{L} first picks a random key k^- in $\{0,1\}^n$, then, he simply issues the query $\mathsf{E}, \mathsf{L_E} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}(p_B)$, receiving $(y := \mathsf{E}_{k_0}(p_B), \mathsf{L}^1)$, where L^1 is either $\mathsf{L}_E(p_B; k_0)$ or $\mathcal{S}^{\mathsf{L}}(k^*, p_B, \mathsf{E}_{k_0}(p_B))$, then, the query $\mathsf{E}, \mathsf{L_E} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}(p_A)$, receiving $(k_1 := \mathsf{E}_{k_0}(p_A), \mathsf{L}^2)$, where L^2 is either $\mathsf{L}_E(p_A; k_0)$ or $\mathcal{S}^{\mathsf{L}}(k^*, p_A, \mathsf{E}_{k_0}(p_A))$, and finally, the query $\mathsf{Gen}\text{-}\mathcal{S}(k^-, p_A)$, receiving L^3 which is either $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0)$ or $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k^*)$. ($k^*$ is the random key picked by the challenger in the $\mathsf{q}\text{-}\mathsf{sim}$ Game). The challenger uses $3q_{\mathcal{S}}$ queries and t_{sim} time to answer to the queries made by B^{L}. Then, B^{L} computes $c = y \oplus m$ and collects its leakage $\mathsf{L}_{\oplus}(m, y)$. This takes time $t_{\mathsf{L}(\oplus)}$. Then B^{L} answers $((c, k_1), \mathsf{L})$ to A^{L}, with $\mathsf{L} = (\mathsf{L}^2, \mathsf{L}^1, \mathsf{L}_{\oplus}(m, y), \mathsf{L}^3, k^-)$.
Answering to A^{L} needs at most $3q_{\mathcal{S}}$ queries and $t_{\mathsf{sim}} + t_{\mathsf{L}(\oplus)}$ time.
When A^{L} outputs a bit b', then, B^{L} outputs the same bit $b'' = b'$.
B^{L} runs in time $t - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)} + t_{\$}$ and he does q_{L} leakage queries. We observe that if the 2-simulatability challenger picks the bit $b = 0$, B^{L} is correctly simu-

lating game 0 for A^{L}; otherwise, he correctly simulates Game 1.

Bounding $|\Pr[E_0] - \Pr[E_1]|$ Clearly $\Pr[2\text{-}\mathsf{sim}(\mathsf{B}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1] = \Pr[E_0]$, while, $\Pr[2\text{-}\mathsf{sim}(\mathsf{B}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1] = \Pr[E_1]$.
Since E has $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_{\mathsf{L}}, t - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)}, \epsilon_{2\text{-}\mathsf{sim}})$ simulatable leakage and B^{L} is a $(q_{\mathsf{L}}, t - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)})$ 2-simulatability adversary we have that

$$\left|\Pr[2\text{-}\mathsf{sim}(\mathsf{B}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1] - \Pr[2\text{-}\mathsf{sim}(\mathsf{B}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 1) = 1]\right| \leq \epsilon_{2\text{-}\mathsf{sim}}$$

thus $|\Pr[E_0] - \Pr[E_1]| \leq \epsilon_{2\text{-}\mathsf{sim}}$.

Game 2 Let Game 2 be the game where the adversary A^{L} is going against scheme $\mathsf{PSV}s^I_{k_0}$, choosing the message m and receiving the ciphertext c, the rekeyed key k_1 and the leakage $\mathsf{L}_{\mathsf{enc}s^I}(m; k_0)$ for a random key k_0.
Let E_2 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Transition between Game 1 and 2 We build a $(2, t - 3t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)} - t_{\$})$-adversary C against the PRF E based on A^{L}.

The $(2, t - q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} - 3t_{\mathcal{S}} - t_{\mathsf{L}(\oplus)} - 2t_{\$})$-$\mathsf{PRF}$ **adversary** C C is playing the PRF game against an oracle, which is either implemented with the PRF E_{k^*}, where k^* is a random key, or a random function f. He is based on the adversary A^{L} and he has to simulate either Game 0 or Game 1 for this adversary A^{L}. At the start of the game, C chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$, which he gives to A^{L}. Moreover, C picks two random key k_0, k^-.
When A^{L} does a leakage query on input $(x; k)$, C computes the leakage $\mathsf{L}_{\mathsf{E}}(x; k)$. It takes time $t_{\mathsf{L}(\mathsf{E})}$ (A^{L} may do at most q_{L} such queries).
When A^{L} asks his query on input m, C first picks a random key k^- in $\{0,1\}^n$, then, he simply queries his oracle on inputs p_B and p_A, receiving y and k_1 respectively. Then, he computes $c = y \oplus m$. Finally, he simulates the leakages $\mathcal{S}^{\mathsf{L}}(k_0, p_B, y)$, $\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1)$, $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0)$ and $\mathsf{L}_{\oplus}(m, y)$ and he answers $((c, k_1), (\mathcal{S}^{\mathsf{L}}(k_0, p_B, y), \mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1), \mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), \mathsf{L}_{\oplus}(m, y), k^-)$ to A^{L}. This takes $3t_{\mathcal{S}} + t_{\mathsf{L}(\oplus)} + 2t_{\$}$ time.
When A^{L} outputs a bit b', C outputs the same bit $b'' = b'$.
C runs in time $t - q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} - 3t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)} - 2t_{\$}$ and does 2 queries to his oracle.
We observe, that, if the oracle is implemented with $\mathsf{E}_{k_0}(\cdot)$, A^{L} is playing Game 1; otherwise, Game 2.
We observe that if the 2-simulatability challenger picks the bit $b = 0$, B^{L} is correctly simulating game 1 for A^{L}; otherwise, he correctly simulates Game 2.

Bounding $|\Pr[E_1] - \Pr[E_2]|$ Clearly $\Pr[E_1] = \Pr[\mathsf{C}^{\mathsf{E}_{k^*}(\cdot)} \Rightarrow 1]$, while, $\Pr[E_2] = \Pr[\mathsf{C}^{\mathsf{f}(\cdot)} \Rightarrow 1]$.
Since E is a $(2, t - q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} - 3t_{\mathcal{S}} - t_{\mathsf{L}(\oplus)} - 2t_{\$}), \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ C is a $(2, t - q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} - 3t_{\mathcal{S}} - t_{\mathsf{L}(\oplus)} - 2t_{\$}))$-$\mathsf{PRF}$ adversary we have that

$$\left|\Pr[\mathsf{C}^{\mathsf{E}_{k^*}(\cdot)} \Rightarrow 1] - \Pr[\mathsf{C}^{\mathsf{f}(\cdot)} \Rightarrow 1]\right| \leq \epsilon_{\mathsf{PRF}}$$

thus $|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{PRF}}$.

Bounding $\Pr[E_0]$ This concludes the proof since we can bound:

$$\left|\Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSVs}_{k_0}(m)) \Rightarrow 1] - \Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSVs}^{I}_{k_0}(m)) \Rightarrow 1]\right| =$$
$$|\Pr[E_0] - \Pr[E_2]| \leq \epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}}$$

We are able to prove the previous lemma, even if the adversary A^{L} outputs two plaintexts $(m^{*,0}, m^{*,1})$ of the same length, and only the first (or the second) is encrypted in all games (that is, Game 0,1 and 2) of the proof. This allow us to reduce the eavesdropper security with leakage of $\mathsf{PSV}s$ to that of $\mathsf{PSV}s^I$:

$$\begin{aligned}
&\Pr[\mathsf{A} \text{ wins } \mathsf{EavL2s}_{\mathsf{PSVs}}] = \\
&\quad \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}} \Rightarrow 1 | b = 1] \Pr[b = 1] + \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}} \Rightarrow 0 | b = 0] \Pr[b = 0] = \\
&\quad\quad \frac{1}{2} \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}} \Rightarrow 1 | b = 1] + \frac{1}{2} \left(1 - \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}} \Rightarrow 1 | b = 0]\right) = \\
&\quad\quad\quad \frac{1}{2} \left(1 + \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}} \Rightarrow 1 | b = 1] - \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}} \Rightarrow 1 | b = 0]\right) = \\
&\frac{1}{2} \left(1 + \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}^I} \Rightarrow 1 | b = 1] - \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}^I} \Rightarrow 1 | b = 0] + 2\epsilon_{\mathsf{2\text{-}sim}} + 2\epsilon_{\mathsf{PRF}}\right) = \\
&\quad \epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}} + \frac{1}{2} \left(\Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}^I} \Rightarrow 1 | b = 1] + \Pr[\mathsf{A}^{\mathsf{L}}_{\mathsf{PSVs}^I} \Rightarrow 0 | b = 0]\right) = \\
&\quad\quad\quad \epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}} + \Pr[\mathsf{A} \text{ wins } \mathsf{EavL2s}_{\mathsf{PSVs}^I}]
\end{aligned}$$

where the second to last equality is given by Lemma 1 with $(m^{*,0}, m^{*,1}) \leftarrow \mathsf{A}^{\mathsf{L}}$, as just discussed.

Discussion of $\Pr[\mathsf{A}$ **wins** $\mathsf{EavL2s}_{\mathsf{PSVs}^I}]$ Since in the proof, we are going to reduce the CPAL2 (and the CCAL2) security of CONCRETE, it is worth to discuss the probability $\Pr[\mathsf{A} \text{ wins } \mathsf{EavL2s}_{\mathsf{PSVs}^I}]$:
First, we observe that it is not clear what this advantage means. In particular, it seems difficult to find a reduction from this advantage to an hard mathematical or physical problem. (Although the leakage problem is less studied than the PRF problem for block cipher, it is the same situation as for AES. There is no proof that AES is a good PRF based on an hard assumption, but the problem is well studied and AES is believed to be a good PRF).
Second, we do not think that this probability may be negligible at the present moment.
Third, we believe that this probability may be evaluated by laboratories and it is much easier to evaluate the probability of success for a single block than for a complete message.
Fourth, understand the leakage of a single block is an easier and more realistic target for research.
Having considered all the pros and cons, we believe that our approach to reduce

the CPAL2 (and CCAL2) security of the whole scheme to the EavL2s security of a single ideal block is the most promising path to have some guarantees of security (same approach as [37,33,11,13,10]).

Multiple block PSV After having proved the security for the single block case, we are going to consider the whole PSV encryption scheme. Again, we define an idealized scheme PSV^I, where the values $k_1, ..., k_l$ and $y_1, ..., y_l$ are randomly picked and not obtained via the PRF E (see for more details Tab. 9 [Similarly to what we did for PSVs we add a random key, k_0, for composability]).

PSV its ideal counterpart PSVs^I	
Gen:	Gen^I:
$k_0 \leftarrow \{0,1\}^n$, $k_1 = \mathsf{E}_{k_0}(p_A)$	$k_1, k_0 \xleftarrow{\$} \{0,1\}^n$
$\mathsf{enc}_{k_1}(m)$:	$\mathsf{enc}^I_{k_1}(m)$:
Pad $m = (m_1, ..., m_l)$ with	Pad $m = (m_1, ..., m_l)$ with
$\lvert m_1\rvert = ... = \lvert m_i\rvert = n$	with $\lvert m_1\rvert = ... = \lvert m_i\rvert = n$
and $\lvert m_l\rvert \leq n$	and $\lvert m_l\rvert \leq n$
For $i = 1, ...l-1$	For $i = 1, ...l-1$
$y_i = \mathsf{E}_{k_i}(p_B)$	$y_i \xleftarrow{\$} \{0,1\}^n$
$c_i = y_i \oplus m_i$	$c_i = y_i \oplus m_i$
$k_{i+1} = \mathsf{E}_{k_i}(p_A)$	$k_{i+1} \xleftarrow{\$} \{0,1\}^n$
$y_l = \mathsf{E}_{k_l}(p_B)$	$y_l \xleftarrow{\$} \{0,1\}^n$
$c_l = \pi_{\lvert m_l\rvert}(y_l) \oplus m_l$	$c_l = \pi_{\lvert m_l\rvert}(y_l) \oplus m_l$
Return $c = (c_1, ..., c_l)$	Return $c = (c_1, ..., c_l)$
$\mathsf{dec}_{k_1}(c)$ proceeds in the natural way.	dec^I proceeds in the natural way.
The leakage resulting from Gen is $\mathsf{L}_{\mathsf{E}}(p_A; k_0)$.	The leakage resulting from Gen^I is $\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1)$.
The leakage resulting from $\mathsf{enc}_{k_1}(m)$ is defined as $\mathsf{L}_{\mathsf{enc}}(m; k_1) := (\mathsf{L}_E(p_B; k_1), \mathsf{L}_{\oplus}(m_1, y_1), \mathsf{L}_E(p_A; k_1), ..., \mathsf{L}_E(p_A; k_{l-1}), \mathsf{L}_E(p_B; k_l), \mathsf{L}_{\oplus}(m_l, \pi_{\lvert m_l\rvert}(y_l)))$.	The leakage resulting from $\mathsf{enc}^I(m)$ is defined as $\mathsf{L}_{\mathsf{enc}^I}(m; k_1) := (\mathcal{S}^{\mathsf{L}}(k_1, p_B, y_1), \mathsf{L}_{\oplus}(m_1, y_1), \mathcal{S}^{\mathsf{L}}(k_1, p_A, k_2), ..., \mathcal{S}^{\mathsf{L}}(k_{l-1}, p_A, k_l), \mathcal{S}^{\mathsf{L}}(k_l, p_B, y_l), \mathsf{L}_{\oplus}(m_l, \pi_{\lvert m_l\rvert}(y_l)))$.

Table 9. PSVs: the single block variant of PSV. The adversary receives the leakage given by Gen and enc at the same moment.

Similarly to what we did in the single block case, we are able to relate the probability that the an adversary wins the EavL2s game when is playing against PSV with the same probability when he is playing against the idealized version, with the following lemma (inspired by Pereira et al. [33]):

Lemma 2. *Let* $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \longmapsto \{0,1\}^n$ *be a* $(2, t - q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} - q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} - t' - 3t_{\mathcal{S}} - t_{\mathsf{L}(\oplus)} - t_{\$}), \epsilon_{\mathsf{PRF}})$-PRF *whose implementation has a leakage function* L *which has* $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_{\mathsf{L}}, t - t' - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)}, \epsilon_{\mathsf{2\text{-}sim}})$*-2-simulatable leakage and let* $\mathcal{S}^{\mathsf{L}}$ *be an appropriate* $(q_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded leakage simulator. Then, if* PSV *takes messages at*

most L bits long, for every $m, p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$, and every (q_{L}, t) adversary A^{L}, the following holds:

$$\left|\Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSV}_{k_0}(m)) \Rightarrow 1] - \Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSV}^I_{k_0}(m)) \Rightarrow 1]\right| \leq$$

$$L\left(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}}\right)$$

where t_{sim} is the time needed to relay the content of the two $\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}$ and the $\mathsf{Gen}\text{-}\mathcal{S}$ query, $t_{\mathsf{L}(\mathsf{E})}$ is the time needed to collect the leakage of a call to E, $t_{\mathsf{L}(\oplus)}$ is the time needed to collect the leakage of XOR of 2 blocks, $t_{\mathcal{S}}$ is the time needed to simulate one leakage query, $t_{\$}$ is the time needed to collect a random block and t' is

$$(L-1)t_{\mathsf{L}(\oplus)} + \max_{i=1,\ldots,L-1}\left[(2i-1)t_{\$} + (2i-3)t_{\mathcal{S}} + (2(L-i)-1)(t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})})\right].$$

Proof. We uses a sequence of games:
Game 0 Let Game 0 be the game where the adversary $\mathsf{A}^{\mathsf{L}}(p_A, p_B)$ is going against scheme PSV_{k_1} choosing the message $m = (m_1, ..., m_l)$, with $|m_1| = ... = |m_{l-1}| = n$, $|m_l| \leq n$ and $l \leq L$, and receiving the ciphertext $c = (c_1, ..., c_l)$ and the leakage $\mathsf{L}_{\mathsf{enc}}(m; k_1)$ for a random key k_1. At the start of the game he is given the constants p_A, p_B. (It is Game 0 of Lemma 1 with a bigger message). The adversary A^{L} is granted q_{L} queries to the leakage oracle.
Let E_0 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Game 1 Let Game 1 be the game where the adversary $\mathsf{A}^{\mathsf{L}}(p_A, p_B)$ is going against scheme $\mathsf{PSV}^I_{k_1}$ choosing the message $m = (m_1, ..., m_l)$, with $|m_1| = ... = |m_{l-1}| = n$, $|m_l| \leq n$ and $l \leq L$, and receiving the ciphertext $c = (c_1, ..., c_l)$ and the leakage $\mathsf{L}_{\mathsf{enc}^I}(m; k_1)$ for a random key k_1. At the start of the game he is given the constants p_A, p_B. The adversary A^{L} is granted to q_{L} queries to the leakage oracle.
Let E_1 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Transition between Game 0 and Game 1 We use at most $L+1$ games Game 1^i with $i = 0, ..., L$ to bound $|\Pr[E_0] - \Pr[E_1]|$.

Game 1^i In this game the adversary A is playing against the following scheme. The first i blocks are computed as $\mathsf{PSV}^I_{k_1}$ obtaining $(c_1, ..., c_i, k_{i+1})$ [if $i = l$, k_{i+1} is not computed], while last $l - i$ blocks are computed as $\mathsf{PSV}_{k_{i+1}}(m_{i+1}, ..., m_l)$. In detail: given a key k_1, for the first i block of plaintext, the corresponding ciphertext block is obtained via $(c_i, k_i + 1) \leftarrow \mathsf{PSV}s^I_{k_i}(m_i)$ with the leakage $l_{\mathsf{encs}^I}(m_i; k_i)$, while for the last $l - i$ blocks of plaintext, the corresponding ciphertext block are obtained via $(c_i, k_i + 1) \leftarrow \mathsf{PSV}s^I_{k_i}(m_i)$ with the leakage $l_{\mathsf{encs}^I}(m_i; k_i)$ [we observe that, for $i = 1$, k_0 is obtained via leakage as k^- in the leakage of PSVs^I]. When the last block m_l is processed, the key k_{l+1} is not computed as well as its corresponding leakage (either $\mathsf{L}_{\mathsf{E}}(p_A; k_l)$ or $\mathcal{S}^{\mathsf{L}}(k_l, p_A, k_{l+1})$).
Let E^i_i be the event that A^{L} outputs 1 at the end of the game.

We observe that Game 1^0 is Game 0, while Game 1^L is Game 1.

Transition between Game 1^i and Game 1^{i-1} We build an adversary $(q_{\mathsf{L}} + (2L-3)q_{\mathcal{S}}, t')$ $\mathsf{B}_i^{\mathsf{L}}$ who has to distinguish $\mathsf{PSV}s$ from $\mathsf{PSV}s^I$, where $t' \leq$

$$(L-1)t_{\mathsf{L}(\oplus)} + \max_{i=1,\ldots,L} \left[(2i-3)t_{\$} + (2i-3)t_{\mathcal{S}} + (2(L-i))(t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})})\right].$$

The $(q_{\mathsf{L}}, t-t')$-adversary $\mathsf{B}_i^{\mathsf{L}}$ $\mathsf{B}_i^{\mathsf{L}}$ has to distinguish if the oracle he is facing is implemented with $\mathsf{PSV}s_{k_i}$ or with $\mathsf{PSV}s_{k_i}^I$. He runs in time t' bounded as before.
At the start of the Game $\mathsf{B}_i^{\mathsf{L}}$ picks 2 constants p_A and p_B in $\{0,1\}^n$ with $p_A \neq p_B$. This takes time at most $2t_{\$}$.
When A^{L} does a leakage query on input $(u; w)$, $\mathsf{B}_i^{\mathsf{L}}$ does the same query and relays the answer to A^{L}. When A^{L} does the challenge query on input $m = (m_1, ..., m_l)$, $\mathsf{B}_i^{\mathsf{L}}$, first asks its oracle the encryption of m_i receiving (y, z) with, as well, the leakage $(\mathsf{L}^1, \mathsf{L}^2, \mathsf{L}_{\oplus}(m_i, y), \mathsf{L}^3, k^-)$ where L^1, L^2 and L^3 are respectively either $\mathsf{L}_{\mathsf{E}}(p_A; k_i)$, $\mathsf{L}_{\mathsf{E}}(p_B; k_i)$ and $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k_i)$ or $\mathcal{S}^{\mathsf{L}}(k_i, p_A, z)$ $\mathcal{S}^{\mathsf{L}}(k_i, p_B, y)$ and $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k_i)$.
First, B_i sets $k_{i-1} = k^-$, then, he picks $i-2$ keys $k_1, ..., k_{i-2}$. Moreover, he picks $i-1$ random values $y_1, ..., y_{i-1}$ which he then XORes to $m_1, ..., m_{i-1}$ obtaining $c_1, ..., c_{i-1}$. After that, he computes the leakages $\mathcal{S}^{\mathsf{L}}(k_1, p_B, y_1), \mathsf{L}_{\oplus}(y_1, m_1), \mathcal{S}^{\mathsf{L}}(k_1, p_A, k_2), ...,$ $\mathcal{S}^{\mathsf{L}}(k_{i-2}, p_A, k_{i-1}), \mathcal{S}^{\mathsf{L}}(k_{i-1}, p_B, y_{i-1}), \mathsf{L}_{\oplus}(y_{i-1}, m_{i-1})$ (the leakage $\mathcal{S}^{\mathsf{L}}(k_{i-1}, p_A, k_i)$ has already been obtained via the oracle query as L^3). This takes at most $(2i-3)t_{\$} + (2i-3)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)}$ time and at most $(2i-3)q_{\mathcal{S}}$ queries.
Second, $\mathsf{B}_i^{\mathsf{L}}$ sets $k_{i+1} = z$, from which he is able to compute all the values y_j and c_j, for $j = i+1, ..., l$, and k_h, for $h = i+2, ..., l-1$, with $y_j = \mathsf{E}_{k_j}(p_B)$ and $k_h = \mathsf{E}_{k_{h-1}}(p_A)$ and $c_j = y_j \oplus m_j$ (if $j = l$, $c_l = \pi_{|m_l|}(y_l) \oplus m_l$), as well with the leakage $\mathsf{L}_{\mathsf{E}}(p_B; k_j)$, $\mathsf{L}_{\mathsf{E}}(p_A; k_{h-1})$ and $\mathsf{L}_{\oplus}(y_j, m_j)$. Computing the last blocks needs at most $(2(l-i)-1)(t_{\mathsf{E}}+t_{\mathsf{L}(\mathsf{E})})+(l-i)t_{\mathsf{L}(\oplus)} \leq (2(L-i)-1)(t_{\mathsf{E}}+t_{\mathsf{L}(\mathsf{E})})+(L-i)t_{\mathsf{L}(\oplus)}$ time.
At the end, $\mathsf{B}_i^{\mathsf{L}}$ answers $(c, [\mathcal{S}^{\mathsf{L}}(k_1, p_B, y_1), \mathsf{L}_{\oplus}(y_1, m_1), \mathcal{S}^{\mathsf{L}}(k_1, p_A, k_2), ..., \mathcal{S}^{\mathsf{L}}(k_{i-2}, p_A, k_{i-1}),$ $\mathcal{S}^{\mathsf{L}}(k_{i-1}, p_B, y_{i-1}), \mathsf{L}_{\oplus}(y_{i-1}, m_{i-1}), \mathsf{L}^3, \mathsf{L}^2, \mathsf{L}_{\oplus}(y_i, m_i), \mathsf{L}^1, \mathsf{L}_{\mathsf{E}}(p_B; k_{i+1}), \mathsf{L}_{\oplus}(y_{i+1}, m_{i+1}), \mathsf{L}_{\mathsf{E}}(p_A; k_{i+1}), ..., \mathsf{L}_{\mathsf{E}}(p$ $\mathsf{L}_{\mathsf{E}}(p_B; k_1), \mathsf{L}_{\oplus}(y_1, m_1)])$ to A^{L}, with $c = (c_1, ..., c_l)$.
When A^{L} outputs a bit b', $\mathsf{B}_i^{\mathsf{L}}$ outputs the same bit $b'' = b'$.
Thus, $\mathsf{B}_i^{\mathsf{L}}$ needs at most time

$$(2i-1)t_{\$} + (2i-3)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)} + (2(l-i)-1)(t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})}) + (l-i)t_{\mathsf{L}(\oplus)} \leq$$

$$(L-1)t_{\mathsf{L}(\oplus)} + (2i-1)t_{\$} + (2i-3)t_{\mathcal{S}} + (2(L-i)-1)(t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})}) \leq$$

$$(L-1)t_{\mathsf{L}(\oplus)} + \max_{i=1,\ldots,L-1} \left[(2i-1)t_{\$} + (2i-3)t_{\mathcal{S}} + (2(L-i)-1)(t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})})\right]$$

and at most $(2L-3)q_{\mathcal{S}}$ queries and q_{L} queries.
We observe that, if the oracle $\mathsf{B}_i^{\mathsf{L}}$ is facing, is implemented with $\mathsf{PSV}s$, $\mathsf{B}_i^{\mathsf{L}}$ correctly simulate Game 1^{i-1} for A^{L}; otherwise, Game 1^i.

Bounding $\left|\Pr[E_1^{i-1}] - \Pr[E_1^i]\right|$ We observe that A is playing the $\mathsf{EavL2s}$ game. Now, we can apply Lemma 1 since adversary $\mathsf{B}_i^{\mathsf{L}}$ is running in time bounded

by $t - t'$ and $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \longmapsto \{0,1\}^n$ is a $(2, t - q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} - t' - 3t_{\mathcal{S}} - t_{\mathsf{L}(\oplus)} - t_{\$}), \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$, whose implementation has a leakage function L which has $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_{\mathsf{L}}, t - t' - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)}, \epsilon_{\mathsf{2\text{-}sim}})$-2-simulatable leakage and where $\mathcal{S}^{\mathsf{L}}$ is an appropriate $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Thus, we can bound,

$$\left|\Pr[E_1^{i-1}] - \Pr[E_1^i]\right| \leq \epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}}$$

Bounding $|\Pr[E_1] - \Pr[E_0]|$ Iterating the previous bound, we conclude the proof:

$$|\Pr[E_1] - \Pr[E_0]| = \left|\Pr[E_1^l] - \Pr[E_1^0]\right| = \sum_{i=1}^{l} \left|\Pr[E_1^{i-1}] - \Pr[E_1^i]\right| \leq$$

$$\sum_{i=1}^{L} (\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}}) \leq L\,(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}})\,.$$

Consequently, to bound the EavL (see Tab. 8) security of PSV, we have to bound the same security for PSV^I.

Lemma 3. *If* PSVs^I *is* $(q_{\mathsf{L}}, t+t', \epsilon)$-$\mathsf{EavL}$ *secure, and if* PSV^I *encrypts messages at most* L *block long, then,* PSV^I *is* $(t, L\epsilon)$-EavL *secure, if it encrypts messages at most* L *blocks long. with* $t' = (2L-5)t_{\$} + (2L-5)t_{\mathcal{S}} + (L-1)t_{\mathsf{L}(\oplus)}$

Proof. Consider an adversary A^{L} playing the EavL game against PSV^I. During this game, he outputs two messages $m^0 = (m_1^0, ..., m_l^0), m^1 = (m_1^1, ..., m_l^1)$ of the same size ($|m^0| = |m^1| \leq Ln$), receiving the encryption of one of them ($c \leftarrow \mathsf{PSV}^I_{k_0}(m^b)$ where b is a secret bit picked uniformly at random). A^{L} has to guess this bit. We use a sequence of games:

Game 0 Let Game 0 be the EavL game when $b = 0$. Let E_0 be the event that the bit, output by A^{L} at the end of the game, is 1.

Game 1 Let Game 1 be the EavL game when $b = 1$. Let E_1 be the event that the bit, output by A^{L} at the end of the game, is 1.

Transition between Game 0 and Game 1 We build a sequence of L games Game 1^i with $i = 0, ..., L$ to bound $|\Pr[E_0] - \Pr[E-1]|$.

Game 1^i Let Game 1^i be the EavL game when the message encrypted is $m^* = (m_1^1, ..., m_i^1, m_{i+1}^0, ..., m_l^0)$ (that is, the first $i-1$ block of the message encrypted are those of m^1 while the last are those from m^0). Let E_1^i be the event that the bit, output by A^{L} at the end of the game, is 1. We observe that Game 0 is Game 1^0, while Game 1 is Game 1^L.

Transition between Game 1^{i-1} and Game 1^i. We build a $(q_{\mathsf{L}}, t-t')$-$\mathsf{EavL2s}$ adversary $\mathsf{B}_i^{\mathsf{L}}$ to bound $\left|\Pr[E_1^{i-1}] - \Pr[E_1^i]\right|$.

The $(q_{\mathsf{L}}, t-t')$**-adversary** $\mathsf{B}_i^{\mathsf{L}}$ $\mathsf{B}_i^{\mathsf{L}}$ has to distinguish if the oracle he is facing, which is implemented with $\mathsf{PSV}s^I$, is encrypting either m_i^0 or m_i^1.
At the start of the Game, the oracle, which $\mathsf{B}_i^{\mathsf{L}}$ faces, picks 2 constants p_A and p_B in $\{0,1\}^n$ with $p_A \neq p_B$. Moreover, it picks two random keys k_i, k^- in $\{0,1\}^n$ which it keeps secret.
When A^{L} does a leakage query on input $(u;w)$, $\mathsf{B}_i^{\mathsf{L}}$ does the same query and relays the answer to A. When A^{L} does the EavL query on input (m^0, m^1) with $m^j = (m_1^j, ..., m_l^j)$, $|m_1^j| = ... = |m_{l-1}^j| = n$ and $|m_l^j| \leq n$, for $j = 0, 1$ $\mathsf{B}_i^{\mathsf{L}}$, first challenges his oracle on input (m_i^0, m_i^1) receiving (y, z) with as well the leakage $(\mathsf{L}^1, \mathsf{L}^2, \mathsf{L}_{\oplus}(m_i^b, y), \mathsf{L}^3, k^-)$ where L^1, L^2 and L^3 are respectively $\mathcal{S}^{\mathsf{L}}(k_i, p_A, z)$ $\mathcal{S}^{\mathsf{L}}(k_i, p_B, y)$ and $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k_i)$.
First, B_i sets $k_{i-1} = k^-$, then, he picks $i-2$ keys $k_1, ..., k_{i-2}$. Moreover, he picks $i-1$ random values $y_1, ..., y_{i-1}$ which he then XORes to $m_1^1, ..., m_{i-1}^1$ obtaining $c_1, ..., c_{i-1}$. After that, he computes the leakages $\mathcal{S}^{\mathsf{L}}(k_1, p_B, y_1), \mathsf{L}_{\oplus}(y_1, m_1^1), \mathcal{S}^{\mathsf{L}}(k_1, p_A, k_2), ...,$ $\mathcal{S}^{\mathsf{L}}(k_{i-2}, p_A, k_{i-1}), \mathcal{S}^{\mathsf{L}}(k_{i-1}, p_B, y_{i-1})$ and $\mathsf{L}_{\oplus}(y_{i-1}, m_{i-1}^1)$ (the leakage $\mathcal{S}^{\mathsf{L}}(k_{i-1}, p_A, k_i)$ has already been obtained via the oracle query as L^3). This takes at most $(2i-3)t_{\$} + (2i-1)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)}$ time and at most $(2i-3)q_{\mathcal{S}}$ queries.
Second, $\mathsf{B}_i^{\mathsf{L}}$ sets $k_{i+1} = z$, then, he picks $l-i-2 \leq L-i-2$ keys $k_{i+2}, ..., k_l$. Moreover, he picks $l-i \leq L-i$ random values $y_{i+1}, ..., y_l$ which he then XORes to $m_{i+1}^0, ..., m_l^0$ obtaining $c_{i+1}, ..., c_l$ (for the last block, $c_l = \pi_{|m_l^0|}(y_l) \oplus m_l^0$). After that, he computes the leakages $\mathcal{S}^{\mathsf{L}}(k_{i+1}, p_B, y_{i+1}), \mathsf{L}_{\oplus}(y_{i+1}, m_{i+1}^0), \mathcal{S}^{\mathsf{L}}(k_{i+1}, p_A, k_{i+2}), ...,$ $\mathcal{S}^{\mathsf{L}}(k_{l-1}, p_A, k_l), \mathcal{S}^{\mathsf{L}}(k_l, p_B, y_{i-1}), \mathsf{L}_{\oplus}(y_l, m_l^0)$ [because the leakage $\mathcal{S}^{\mathsf{L}}(k_i, p_A, k_{i+1})$ has already been obtained via the oracle query as L^2]. This takes at most $2(l-i-1)t_{\$} + 2(l-i-1)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)} \leq 2(L-i-1)t_{\$} + 2(L-i-1)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)}$ time and at most $2(l-i-1)q_{\mathcal{S}} \leq 2(L-i-1)q_{\mathcal{S}}$ queries.
At the end, $\mathsf{B}_i^{\mathsf{L}}$ answers $(c = (c_1, ..., c_l), (\mathcal{S}^{\mathsf{L}}(k_1, p_B, y_1), \mathsf{L}_{\oplus}(y_1, m_1^1), \mathcal{S}^{\mathsf{L}}(k_1, p_A, k_2),$ $..., \mathcal{S}^{\mathsf{L}}(k_{i-2}, p_A, k_{i-1}), \mathcal{S}^{\mathsf{L}}(k_{i-1}, p_B, y_{i-1}), \mathsf{L}_{\oplus}(y_{i-1}, m_{i-1}^1), \mathsf{L}^3, \mathsf{L}^2, \mathsf{L}_{\oplus}(y_i, m_i^b), \mathsf{L}^1, \mathcal{S}^{\mathsf{L}}(k_{i+1}, p_B, y_{i+1}),$ $\mathsf{L}_{\oplus}(y_{i+1}, m_{i+1}^0), \mathcal{S}^{\mathsf{L}}(k_{i+1}, p_A, k_{i+2}), ..., \mathcal{S}^{\mathsf{L}}(k_{l-1}, p_A, k_l), \mathcal{S}^{\mathsf{L}}(k_l, p_B, y_{i-1}), \mathsf{L}_{\oplus}(y_l, m_l^0)))$ to A^{L}.
When A^{L} outputs a bit b', $\mathsf{B}_i^{\mathsf{L}}$ outputs the same bit $b'' = b'$.
Thus, $\mathsf{B}_i^{\mathsf{L}}$ needs at most

$$(2i-3)t_{\$} + (2i-1)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)} + 2(l-i-1)t_{\$} + 2(l-i-1)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)} \leq$$

$$(2i-3)t_{\$} + (2i-1)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)} + 2(L-i-1)t_{\$} + 2(L-i-1)t_{\mathcal{S}} + (i-1)t_{\mathsf{L}(\oplus)} \leq$$

$$(2L-5)t_{\$} + (2L-5)t_{\mathcal{S}} + (L-1)t_{\mathsf{L}(\oplus)} = t'$$

time to simulate the challenge query for A^{L} and at most

$$(2i-3)q_{\mathcal{S}} + 2(l-i-1)q_{\mathcal{S}} \leq (2i-3)q_{\mathcal{S}} + 2(L-i-1)q_{\mathcal{S}} \leq (2L-5)q_{\mathcal{S}}$$

queries to the simulator.
We observe that if the message, encrypted by the oracle $\mathsf{PSV}s^I$, which $\mathsf{B}_i^{\mathsf{L}}$ is facing, is $m^{0,i}$, $\mathsf{B}_i^{\mathsf{L}}$ correctly simulate Game 1^{i-1} for A^{L}; otherwise, Game 1^i.

Bounding $\left|\Pr[E_1^{i-1}] - \Pr[E_1^i]\right|$ We observe that the probability $\mathsf{B}_i^{\mathsf{L}}$ wins the previous $\mathsf{EavL2s}$ game against PSVs^I is

$$\Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 0|b=0]\Pr[b=0] + \Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 1|b=1]\Pr[b=1] =$$

$$\frac{1}{2}\left(1 - \Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 1|b=0] + \Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 1|b=1]\right) \leq \frac{1}{2} - \frac{|\Pr[E_1^{i-1}] - \Pr[E_1^i]|}{2}.$$

Since $\mathsf{B}_i^{\mathsf{L}}$ is a $(q_{\mathsf{L}}, t-t')$-adversary and since $\mathsf{PSV}s^I$ is $(q^{\mathsf{L}}, t-t', \epsilon)$-$\mathsf{EavL2s}$-secure, we can bound

$$\left|\Pr[E_1^{i-1}] - \Pr[E_1^i]\right| \leq 2\epsilon.$$

Bounding $|\Pr[E_0] - \Pr[E_1]|$ Iterating the previous bound, we conclude the proof:

$$|\Pr[E_0] - \Pr[E_1]| = \left|\Pr[E_1^0] - \Pr[E_1^L]\right| \leq$$

$$\sum_{i=1}^{L} \left|\Pr[E_1^{i-1}] - \Pr[E_1^i]\right| \leq 2L\epsilon$$

Bounding the $\mathsf{EavL2s}$ **security of** PSV^I Now we can compute the probability that A^{L} wins the $\mathsf{EavL2s}$ game against PSV^I:

$$\Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 0|b=0]\Pr[b=0] + \Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 1|b=1]\Pr[b=1] =$$

$$\frac{1}{2}\left(1 - \Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 1|b=0] + \Pr[\mathsf{A}^{\mathsf{L}} \Rightarrow 1|b=1]\right) \leq$$

$$\frac{1}{2} - \frac{|\Pr[E_0] - \Pr[E_1]|}{2} \leq \frac{1}{2} + \frac{2L\epsilon}{2}.$$

which concludes the proof, showing that PSV^I is $(q^{\mathsf{L}}, t-t', L\epsilon) - \mathsf{EavL2s}$-secure.

We observe that for PSV^I, k_0 can be given as leakage, without any security problem, (it is k^- of the first block.

Security of the first block c_0 We cannot extend the previous analysis to the complete $\mathsf{CONCRETE}$ because, although the first block c_0 can be seen as $\mathsf{PSV}s_{k_0}(0^n)$, there is an additional source of leakage given by $c_{l+1} = \mathsf{F}_k^*(h, k_0)$ (the leak free component protects only its key, not its input). Moreover, the key k_0 is picked uniformly at random by the algorithm (and not via $k_0 = \mathsf{E}_{k_{-1}}(p_A)$), which may leak differently. On the other hand, since c_0 does not depend on the challenge plaintext, we do not care of the $\mathsf{EavL2s}$-security of the first block, but we require only to be able to replace the computation of this first bloc, with an ideal one, which does not affect the security of the following PSV^I.
In order to do this, we have to modify the $\mathsf{q\text{-}sim}$ definition to consider these additional sources of leakage. But, first we have to simulate these leakages $\mathsf{L}_{\mathsf{F}^*}(h, k_0; k)$ and $\mathsf{L}_{\$}(k_0)$.

Definition 31. *A leak free component has $(q, q_{\mathcal{S}'}, t_{\mathcal{S}'}, t)$-indistinguishable leakage if for any (q, t) adversary, there exists a $(q_{\mathcal{S}'}, t_{\mathcal{S}'})$-simulator such that the leakage $\mathsf{L}_{\mathsf{F}^*}(x, y; k)$ of the computation $z \leftarrow \mathsf{F}^*_k(x, y)$ is indistinguishable from the simulated leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(x, y, z, k^*)$ for a random key k^*.*

Since, by hypothesis, the leak free component hides the key it uses, although it is very strong, the previous definition can be assumed given the leak free hypothesis. In particular, we suppose that the leakage of F^* depends only on its inputs and outputs.

Regarding the leakage of k_0, when it is picked uniformly at random, we suppose that its leakage is given by a simulator $\mathcal{S}_\$$ which uses as only input k_0.

Now, we can modify the q−sim game in order to consider the additional source of leakage for k_0, obtaining the q−sim′ game:

Game $\mathsf{q\text{−}sim}'(\mathsf{A}, \mathsf{PRF}, \mathsf{L}, \mathcal{S}, \mathcal{S}_{\mathsf{F}^*}, b)$		
The challenger selects three random keys $k, k^*, k^+ \xleftarrow{\$} \mathcal{K}$ and a random value $w \xleftarrow{\$} \{0,1\}^n$. The output of the game is a bit b' computed by A^{L} based on the challenger responses to a total of at most q adversarial queries of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{E}, \mathsf{L}_{\mathsf{E}}$ \ $\mathsf{E}, \mathcal{S}^{\mathsf{L}}(x)$	$\mathsf{E}_k(x)$, $\mathsf{L}(k, x)$	$\mathsf{E}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{E}_k(x))$
and one query of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}\text{−}\mathcal{S}'()$	$\mathsf{L}_\$(k)$	$\mathsf{L}_\$(k^*)$
and one query of the following type:		
$\mathsf{Key}\text{−}\mathsf{Send}(h)$	$\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(h, k, k^+, w)$	$\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(h, k^*, k^+, w)$

Table 10. The q−sim′ experiment used for the first block

Definition 32. [q-simulatable leakages'] *Let E be a PRF having leakage function L and let F^* be a STPRP having $(q_{\mathcal{S}'}, t_{\mathcal{S}'})$-indistinguishable leakage (see Def. ??). Then E has $(q_{\mathcal{S}}, q'_{\mathcal{S}}, q_{\mathsf{A}}, t_{\mathcal{S}}, t'_{\mathcal{S}}, t_{\mathsf{A}}, \epsilon_{\mathsf{q\text{−}sim}})$ q-simulatable' leakage if there is a $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded simulator $\mathcal{S}^{\mathsf{L}}$ such that, for every $(q_{\mathsf{A}}, t_{\mathsf{A}})$-bounded adversary A^{L}, we have*

$$|\Pr[\mathsf{q\text{−}sim}'(\mathsf{A}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 1) = 1] - \Pr[\mathsf{q\text{−}sim}'(\mathsf{A}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1]| \leq \epsilon_{\mathsf{q\text{−}sim}}.$$

(The q-simulatability' assumption is clearly the q-simulatability assumption [Def. 23 or [33]] with the two modifications: the key k is not generated via E but picked uniformly at random and it is also encrypted using F^*).

Now, we want to study the security of the first block c_0: we introduce a scheme PSVsfb and its ideal counterpart PSVsfb^I.

Lemma 4. *Let* F^* *be a* $(1, t + t'', \epsilon_{\mathsf{STPRP}})$-$\mathsf{STPRP}$*, whose implementation has* $(1, q_{\mathcal{S}'}, t_{\mathcal{S}'}, t + t')$- *indistinguishable leakage, let* E *be a* $(2, t + t''', \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$*, whose implementation has* $(q_{\mathcal{S}}, q'_{\mathcal{S}}, q_{\mathsf{L}}, t_{\mathcal{S}}, t'_{\mathcal{S}}, t + t''', \epsilon_{2\text{-}\mathsf{sim}'})$*-2-simulatable' leakage let then, for any* (q_{L}, t)*-adversary* A^{L}*,*

$$\left|\Pr[\mathsf{A}^{\mathsf{L}}(\mathsf{PSVsfb}) \Rightarrow 1] - \Pr[\mathsf{A}^{\mathsf{L}}(\mathsf{PSVsfb}^I) \Rightarrow 1]\right| \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{2\text{-}\mathsf{sim}'} + \epsilon_{\mathsf{PRF}}$$

where

$$t' = t_{\$} + t_{\mathsf{L}(\$)} + 2t_{\mathsf{E}} + 2t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{F}^*}$$
$$t'' = q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\$} + t_{\mathsf{L}(\$)} + 2t_{\mathsf{E}} + 2t_{\mathsf{L}(\mathsf{E})} + t_{\mathcal{S}'}$$
$$t''' = q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)}$$

$t_{\$}$ *is the time necessary to pick uniformly a random block and* $t_{\mathsf{L}(\$)}$ *is the time needed to collect its leakage,* t_{E} *is the time needed to compute* E *and* $t_{\mathsf{L}(\mathsf{E})}$ *the time to collect its leakage,* t_{F^*} *is the time needed to compute* F^**,* $t_{\mathsf{sim}'}$ *is the time needed to run the* 2*-simulatability' experiment.*

This lemma is Lemma 1 adapted to the situation for the first block, where no message is used, but where there is an additional source of leakage. The proof is a straightforward adaptation of that proof.

Proof. Again we use a sequence of games:

Game 0 Let Game 0 be the game where the adversary $\mathsf{A}^{\mathsf{L}}(p_A, p_B)$ is going against scheme $\mathsf{PSVsfb}_{k_0,k}$, choosing an input h (it is not a message, because it is not encrypted, it is the tweak used by F^*_k), receiving the ciphertext c, the rekeyed key k_1 and the leakage $\mathsf{L}_{\mathsf{encsfb}}(h; k_0, k)$ for two random keys k_0, k. He is given the constants p_A, p_B. (That is, it is an eavesdropper game with leakage where the adversary receives only the output of an execution of the algorithm). The adversary A^{L} is granted to q_{L} queries to the leakage oracle.
Let E_0 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Game 1 Let Game 1 be Game 0 where we have replaced the leakage $\mathsf{L}_{\mathsf{F}^*}(k_0, h; k)$ with $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', k_0, h, d)$ for a random key k' in the leakage of encsfb_{k_0}.
Let E_1 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Transition between Game 0 and 1 We build a $(1, t - t')$-adversary B^{L} against the indistinguishability of the leakage of F^* based on A to bound the absolute difference $|\Pr[E_0] - \Pr[E_1]|$.

The $(1, t')$ **indistinguishability adversary** B^{L} B^{L} has to distinguish if the leakage he obtains is $\mathsf{L}_{\mathsf{F}^*}(k_0, h; k)$ or $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', k_0, h, d)$ for a random key k'. At the start of the game, B^{L} chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ which he gives to A^{L}.
When A^{L} does a leakage query, B^{L} does the same query and relays its answer to A^{L}.

When A^{L} asks to run PSVsfb on input h, B^{L} first picks two random keys k_0, k in $\{0,1\}^n$, then, he simply issues computes $c = y\mathsf{E}_{k_0}(p_B)$, $k_1 = \mathsf{E}_{k_0}(p_A)$ and $d = \mathsf{F}^*_k(k_0, h)$. Then he queries his oracle on input $(k_0, h; k_0)$ receiving a leakage L^1 which is either $\mathsf{L}_{\mathsf{F}^*}(k_0, h; k)$ or $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', k_0, h, d)$. after that, he computes the leakage $\mathsf{L}_{\mathsf{E}}(p_A; k_0)$, $\mathsf{L}_{\mathsf{E}}(p_B; k_0)$ and $Ł_{\$}(k_0)$ answering to A^{L} $((c, k_1), (\mathsf{L}_{\mathsf{E}}(p_A; k_0),$ $\mathsf{L}_{\mathsf{E}}(p_B; k_0)$, $Ł_{\$}(k_0)$, $\mathsf{L}^1))$. Answering to A^{L} needs at most 1 oracle query and $t_{\$} + t_{\mathsf{L}(\$)} + 2t_{\mathsf{E}} + 2t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{F}^*} = t'$ time.
When A^{L} outputs a bit b', then, B^{L} outputs the same bit $b'' = b'$.
B^{L} runs in time $t - t'$ and he does 1 query to his oracle. We observe that if the indistinguishability adversary picks the bit $b = 0$, B^{L} is correctly simulating game 0 for A^{L}; otherwise, he correctly simulates Game 1.

Bounding $|\Pr[E_0] - \Pr[E_1]|$ Clearly $\Pr[\mathsf{B}^{\mathsf{L}_{\mathsf{F}^*}} \Rightarrow 1] = \Pr[E_0]$, while,
$\Pr[\mathsf{B}^{\mathcal{S}'^{\mathsf{f}}{}_{\mathsf{F}^*}} \Rightarrow 1] = \Pr[E_1]$.
Since F^* has $(1, q_{\mathcal{S}'}, t_{\mathcal{S}'}, t - t')$-indistinguishable leakage and B^{L} is a $(1, t - t')$ adversary we have that

$$\left|\Pr[\mathsf{B}^{\mathsf{L}_{\mathsf{F}^*}} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathcal{S}'^{\mathsf{f}}{}_{\mathsf{F}^*}} \Rightarrow 1]\right| = 0$$

thus $\Pr[E_0] = \Pr[E_1]$.

Game 2 It is Game 1 where d is picked uniformly at random in stead of being computed as $d = \mathsf{F}^*_k(k_0, h)$.

The $(1, t - t'')$**-**STPRP **adversary** C C is playing the STPRP game against an oracle, which is either implemented with the STPRP F^*_k, where k is a random key, or a random tweakable permutation f. He is based on the adversary A^{L} and he has to simulate either Game 1 or Game 2 for this adversary A^{L}. At the start of the game, C chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$, which he gives to A^{L}. Moreover, C picks two random keys k_0, k^* and collects the leakage $\mathsf{L}_{\$}(k_0)$.
When A^{L} does a leakage query on input $(x; k)$, C computes the leakage $\mathsf{L}_{\mathsf{E}}(x; k)$. It takes time $t_{\mathsf{L}(\mathsf{E})}$ (A^{L} may do at most q_{L} such queries).
When A^{L} asks to run PSVsfb on input h, C simply computes $c = y = \mathsf{E}_{k_0}(p_B)$ and $k_1 = \mathsf{E}_{k_0}(p_A)$ and collects the leakage $\mathsf{L}_{\mathsf{E}}(p_B; k_0)$ and $\mathsf{L}_{\mathsf{E}}(p_A; k_0)$, then, he simply queries his oracle on inputs (k_0, h), receiving d. Finally, he simulates the leakages $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k^*, k_0, h, d)$ and he answers $((c, k_1), (\mathsf{L}_{\mathsf{E}}(p_B; k_0),$
$\mathsf{L}_{\mathsf{E}}(p_A; k_0)\mathsf{L}_{\$}(k_0), \mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k^*, k_0, h, d)))$ to A^{L}. This takes $2t_{\$} + t_{\mathsf{L}(\$)} + 2t_{\mathsf{E}} + 2t_{\mathsf{L}(\mathsf{E})} + t_{\mathcal{S}'}$ time and 1 oracle query.
When A^{L} outputs a bit b', C outputs the same bit $b'' = b'$.
C does 1 query to his oracle and runs in time bounded by $t - t''$ with

$$t'' = q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\$} + t_{\mathsf{L}(\$)} + 2t_{\mathsf{E}} + 2t_{\mathsf{L}(\mathsf{E})} + t_{\mathcal{S}'}$$

We observe, that, if the oracle is implemented with $\mathsf{F}^*_k(\cdot, \cdot)$, C correctly simulates Game 1 for A^{L}; otherwise, Game 2.

Bounding $|\Pr[E_1] - \Pr[E_2]|$ Clearly $\Pr[E_1] = \Pr[\mathsf{C}^{\mathsf{F}^*_k(\cdot,\cdot)} \Rightarrow 1]$, while, $\Pr[E_2] = \Pr[\mathsf{C}^{\mathsf{f}(\cdot,\cdot)} \Rightarrow 1]$.
Since F^* is a $(1, t - t'', \epsilon_{\mathsf{STPRP}})$-STPRP and C is a $(1, t - t'')$-STPRP adversary we have that

$$\left|\Pr[\mathsf{C}^{\mathsf{F}_k(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathsf{C}^{\mathsf{f}(\cdot,\cdot)} \Rightarrow 1]\right| \leq \epsilon_{\mathsf{STPRP}}$$

thus $|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{STPRP}}$.
(We observe that a $(1, t, \epsilon_{\mathsf{STPRP}})$ is a $(1, t, \epsilon_{\mathsf{PRF}})$ since there is only one query and there is no need to difference between a permutation and a function).

Game 3 Let Game 3 be Game 2 where we have replaced the leakages $(\mathsf{L}_E(p_A; k_0), \mathsf{L}_E(p_B; k_0))$ with $(\mathcal{S}^{\mathsf{L}}(k^*, p_A, k_1), \mathcal{S}^{\mathsf{L}}(k^*, p_B, y))$ for a random key k^* in the leakage of encsfb_{k_0}.
Let E_3 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Transition between Game 2 and 3 We build a $(q_{\mathsf{L}}, t - t''')$-adversary D^{L} against the 2-simulatability' of E based on A^{L} to bound the absolute difference $|\Pr[E_2] - \Pr[E_3]|$.

The $(q_{\mathsf{L}}, t+t''')$-2-simulatability adversary D^{L} D^{L} is playing the 2-simulatability' game (see Sec. 5.4 and Tab. 10) and he has to simulate either Game 2 or Game 3 for A^{L}. At the start of the game, D^{L} chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ which he gives to A^{L}.
When A^{L} does a leakage query, D^{L} does the same query and relays its answer to A^{L}.
When A^{L} asks to execute PSVsfb on input h, B^{L} first picks two random keys k_0, k^+ in $\{0,1\}^n$, then, he simply issues the query $\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}(p_B)$, receiving $(y := \mathsf{E}_{k_0}(p_B), \mathsf{L}^1)$, where L^1 is either $\mathsf{L}_E(p_B; k_0)$ or $\mathcal{S}^{\mathsf{L}}(k^*, p_B, \mathsf{E}_{k_0}(p_B))$, then, the query $\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}(p_A)$, receiving $(k_1 := \mathsf{E}_{k_0}(p_A), \mathsf{L}^2)$, where L^2 is either $\mathsf{L}_E(p_A; k_0)$ or $\mathcal{S}^{\mathsf{L}}(k^*, p_A, \mathsf{E}_{k_0}(p_A))$, after that, the query $\mathsf{Gen}\text{-}\mathcal{S}()$, receiving L^3 which is either $\mathsf{L}_\$(k)$ or $\mathsf{L}_\$(k^*)$ and, finally, after having picked a random w, D^{L} asks the $\mathsf{Key}\text{-}\mathsf{Send}(h)$ query, obtaining L^4 which is either $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(k^+, k_0, h, w)$ or $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(k^+, k^*, h, w)$. ($k^*$ is the random key picked by the challenger in the $\mathsf{q}\text{-}\mathsf{sim}'$ Game). The challenger uses $2q_{\mathcal{S}}$ and $q'_{\mathcal{S}}$ queries and $t_{\mathsf{sim}'}$ time to answer to the queries made by B^{L}. Then, D^{L} computes $c = y \oplus m$ and collects its leakage $\mathsf{L}_\oplus(m, y)$. This takes time $t_{\mathsf{L}(\oplus)}$. Then D^{L} answers $((c, k_1, d), \mathsf{L})$ to A^{L}, with $\mathsf{L} = (\mathsf{L}^2, \mathsf{L}^1, \mathsf{L}_\oplus(m, y), \mathsf{L}^3, \mathsf{L}^4)$. Answering to A^{L} needs at most $3q_{\mathcal{S}}$ queries and $t_{\mathsf{sim}'} + t_{\mathsf{L}(\oplus)}$ time, where $t_{\mathsf{sim}'}$ is the time necessary to run the $2\text{-}\mathsf{sim}'$ experiment.
When A^{L} outputs a bit b', then, B^{L} outputs the same bit $b'' = b'$.
B^{L} runs in time $t - t_{\mathsf{sim}} - t_{\mathsf{L}(\oplus)}$ and he does q_{L} leakage queries.
If the challenger involved in the 2-simulatability' experiment picks the bit $b = 0$, D^{L} correctly simulates Game 2; otherwise, Game 3.

Bounding $|\Pr[E_3] - \Pr[E_2]|$ Clearly $\Pr[2\text{-}\mathsf{sim}'(\mathsf{D}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1] = \Pr[E_2]$, while, $\Pr[2\text{-}\mathsf{sim}'(\mathsf{D}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1] = \Pr[E_3]$.

Since E has $(q_{\mathcal{S}}, q'_{\mathcal{S}}, q_{\mathsf{L}}, t_{\mathcal{S}}, t'_{\mathcal{S}}, t - t_{\mathsf{sim}'} - t_{\mathsf{L}(\oplus)}, \epsilon_{\mathsf{2\text{-}sim}'})$ simulatable' leakage and D^{L} is a $(q_{\mathsf{L}}, t - t_{\mathsf{sim}'} - t_{\mathsf{L}(\oplus)})$ 2-simulatabilityà adversary we have that

$$\left|\Pr[\mathsf{2\text{-}sim}'(\mathsf{D}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1] - \Pr[\mathsf{2\text{-}sim}'(\mathsf{D}^{\mathsf{L}}, \mathsf{E}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 1) = 1]\right| \leq \epsilon_{\mathsf{2\text{-}sim}'}$$

thus $|\Pr[E_2] - \Pr[E_3]| \leq \epsilon_{\mathsf{2\text{-}sim}'}$.

Game 4 Let Game 4 be the game where the adversary A^{L} is going against scheme $\mathsf{PSVsfb}^I_{k_0}$, choosing the message m and receiving the ciphertext (c, d), the rekeyed key k_1 and the leakage $\mathsf{L}_{\mathsf{encsfb}^I}(m; k_0)$ for a random key k_0.
Let E_4 be the probability that the adversary A^{L} outputs 1 at the end of the game.

Transition between Game 3 and 4 We build a $(2, t + q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)})$-adversary EE against the PRF E based on A^{L}.

The $(2, t + q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)})$-$\mathsf{PRF}$ **adversary** EE EE is playing the PRF game against an oracle, which is either implemented with the PRF E_{k^*}, where k^* is a random key, or a random function f. He is based on the adversary A^{L} and he has to simulate either Game 0 or Game 1 for this adversary A^{L}. At the start of the game, EE chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$, which he gives to A^{L}. Moreover, C picks two random keys k_0, k.
When A^{L} does a leakage query on input $(x; k)$, EE computes the leakage $\mathsf{L}_{\mathsf{E}}(x; k)$. It takes time $t_{\mathsf{L}(\mathsf{E})}$ (A^{L} may do at most q_{L} such queries).
When A^{L} asks to run PSVsfb on input h, C first picks a random key k_0 in $\{0,1\}^n$, then, he simply queries his oracle on inputs p_B and p_A, receiving y and k_1 respectively. Then, he sets $c = y$. After that, he picks uniformly at random a value d in $\{0,1\}^n$. Finally, he simulates the leakages $\mathcal{S}^{\mathsf{L}}(k_0, p_B, y)$, $\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1)$ and $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(k, k_0, h, d)$ and computes the leakage $\mathsf{L}_{\$}(k_0)$ and he answers $((c, k_1, d), (\mathcal{S}^{\mathsf{L}}(k_0, p_B, y), \mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1), \mathsf{L}_{\$}(k_0), \mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(k, k_0, h, d))$ to A^{L}. This takes $2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)}$ time.
When A^{L} outputs a bit b', EE outputs the same bit $b'' = b'$.
EE runs in time $t + q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)}$ and does 2 queries to his oracle.
We observe, that, if the oracle is implemented with $\mathsf{E}_{k^*}(\cdot)$, A^{L} is playing Game 3; otherwise, Game 4.

Bounding $|\Pr[E_3] - \Pr[E_4]|$ Clearly $\Pr[E_3] = \Pr[\mathsf{EE}^{\mathsf{E}_{k^*}(\cdot)} \Rightarrow 1]$, while, $\Pr[E_4] = \Pr[\mathsf{EE}^{\mathsf{f}(\cdot)} \Rightarrow 1]$.
Since E is a $(2, t + q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)}, \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ EE is a $(2, t + q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)})$-$\mathsf{PRF}$ adversary we have that

$$\left|\Pr[\mathsf{EE}^{\mathsf{E}_{k^*}(\cdot)} \Rightarrow 1] - \Pr[\mathsf{EE}^{\mathsf{f}(\cdot)} \Rightarrow 1]\right| \leq \epsilon_{\mathsf{PRF}}$$

thus $|\Pr[E_4] - \Pr[E_3]| \leq \epsilon_{\mathsf{PRF}}$.

Bounding $\Pr[E_0]$ This concludes the proof since we can bound:

$$\left|\Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSVs}_{k_0}(h)) \Rightarrow 1] - \Pr[\mathsf{A}^{\mathsf{L}}(m, \mathsf{PSVs}^{I}_{k_0}(h)) \Rightarrow 1]\right| =$$

$$|\Pr[E_4] - \Pr[E_0]| \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{2\text{-}sim'}} + \epsilon_{\mathsf{PRF}}$$

We observe that if we allow the adversaries to choose h after having received c, k_1 and the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0), \mathsf{L}_{\mathsf{E}}(p_A; k_0), \mathsf{L}_{\$}(k_0)$, this does not change anything, since in the $\mathsf{q\text{-}sim'}$ game it is allowed (For the other games, the indstinguishability, the STPRP and the PRF game, it does not change)/

This allow us to, finally, prove the EavL2 security of CONCRETE:

Proposition 1. *Let* F^* *be a* $(1, t + t' + t_2, \epsilon_{\mathsf{STPRP}})$-STPRP*, whose implementation has* $(1, q_{\mathcal{S}'}, t + t' + t_1, t_{\mathcal{S}'})$- *indistinguishable leakage, let* E *be a* $(2, t + max(t' + t_3, t'' + t_4 + t_5), \epsilon_{\mathsf{PRF}})$-PRF*, whose implementation has* $(q_{\mathcal{S}}, q_{\mathsf{L}}, t + t'' + t_4 + t_6, t_{\mathcal{S}}, \epsilon_{\mathsf{2\text{-}sim}})$*-2-simulatable leakage and* $(q_{\mathcal{S}}, q_{\mathcal{S}'}, q_{\mathsf{L}}, t + t' + t_3, t_{\mathcal{S}}, t_{\mathcal{S}'}, \epsilon_{\mathsf{2\text{-}sim'}})$*-2-simulatable' leakage let* $\mathsf{PSV}s^I$ *be* $(q_{\mathsf{L}}, t + t'' + t_7, \epsilon_{\mathsf{EavL2s}})$-EavL2*-secure, then* CONCRETE*, if encrypts at most* L *block messages, is* $(q_{\mathsf{L}}, t, \epsilon)$-EavL2*-secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{2\text{-}sim'}} + (L + 1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{EavL2s}})$$

where

$$t' = t_{\$} + 2Lt_{\mathsf{E}} + 2Lt_{\mathsf{L(E)}} + Lt_{\mathsf{L}(\oplus)} + t_{\mathsf{H}}$$

$$t'' = 3t_{\$} + t_{\mathsf{L}(\$)} + t_{\mathsf{H}} + t_{\mathcal{S}} + t_{\mathcal{S}'}$$

$$t_1 = t_{\$} + t_{\mathsf{L}(\$)} + 2t_{\mathsf{E}} + 2t_{\mathsf{L(E)}} + t_{\mathsf{F}^*}$$

$$t_2 = q_{\mathsf{L}}t_{\mathsf{L(E)}} + 2t_{\$} + t_{\mathsf{L}(\$)} + 2t_{\mathsf{E}} + 2t_{\mathsf{L(E)}} + t_{\mathcal{S}'}$$

$$t_3 = q_{\mathsf{L}}t_{\mathsf{L(E)}} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)}$$

$$t_4 = (L-1)t_{\mathsf{L}(\oplus)} + \max_{i=1,\ldots,L-1}\left[(2i-1)t_{\$} + (2i-3)t_{\mathcal{S}} + (2(L-i)-1)(t_{\mathsf{E}} + t_{\mathsf{L(E)}})\right]$$

$$t_5 = q_{\mathsf{L}}t_{\mathsf{L(E)}} + 3t_{\mathcal{S}} + t_{\mathsf{L}(\oplus)} + 2t_{\$}$$

$$t_6 = q_{\mathsf{L}}t_{\mathsf{L(E)}} + t_{\mathsf{sim}} + t_{\mathsf{L}(\oplus)} + 2t_{\$}$$

$$t_7 = (2L-5)t_{\$} + (2L-5)t_{\mathcal{S}} + (L-1)t_{\mathsf{L}(\oplus)}$$

where $t_{\$}$ *is the time needed to sample a random block,* $t_{\mathsf{L}(\$)}$ *is the time needed to collect the leakage of the random sampling of a block,* t_{E} *the needed to execute* E*,* $t_{\mathsf{L(E)}}$ *the time needed to collect the leakage of* E*,* $t_{\mathsf{L}(\oplus)}$ *the time needed to collect the leakage of the XOR* ($\oplus$) *of two blocks,* t_{H} *is the time needed to execute once the hash function,* t_{F^*} *is the time needed to execute once the* STPRP F^* *and* t_{sim} *is the time needed to relay the content of the two* $\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}$ *and the* $\mathsf{Gen\text{-}}\mathcal{S}$ *query.*

The idea of the proof is first, to replace the computation of c_0 and c_{l+1} with PSVsfb^I (see Tab. 7) and, finally, to use the EavL2 security of PSV.

Proof. **Game 0** It is the standard $\mathsf{EavL2}$ game played by A^{L} against mode $\mathsf{CONCRETE}$. Let E_0 be the probability that A^{L} wins Game 0.

Game 1 It is Game 0, where we have replaced c_0, k_1 and c_{l+1} with random values and the leakages $\mathsf{L_E}(p_B; k_0), \mathsf{L_E}(p_A; k_0)$ and $\mathsf{L}_{\mathsf{F}^*}(k_0, h; k)$ respectively with the simulated leakages $\mathcal{S}^{\mathsf{L}}(k_0, p_B, c_0), \mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1)$ and $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k, h, k_0, c_{l+1})$. Let E_1 be the probability that A^{L} wins Game 1.
We observe that Game 1 is Game 0, where instead of PSVsfb, PSVsfb^I is used in mode $\mathsf{CONCRETE}$.

Transition between Game 0 and Game 1 We build a $(q_{\mathsf{L}}, t + t')$-adversary B^{L} who wants to distinguish PSVsfb from PSVsfb^I based on A^{L}.
The $(q_{\mathsf{L}}, t + t')$-adversary B^{L} B^{L} has to distiguish PSVsfb from PSVsfb^I. At the start of the game, the oracle picks two keys k_0 and k uniformly at random in $\{0,1\}^n$.
When A^{L} does a leakage query, B^{L} does the same query and relays the answer to A^{L}.
When A^{L} outputs his challenge plaintexts (m^0, m^1), with $|m^0| = |m^1| \leq Ln$, B^{L} proceeds as follow: first, he picks uniformly at random a bit $b \leftarrow \{0,1\}$ and he parses the message m^b in $m^b = (m_1^b, ..., m_l^b)$ with $|m_1^b| = ... = |m_{l-1}^b| = n$ and $|m_l^1| \leq n$, then, he runs its oracle to obtain the first part of PSVsfb, that is, (c_0, k_1) and the leakages $(\mathsf{L}^1, \mathsf{L}^2, \mathsf{L}^3)$, where L^1 is either $\mathsf{L_E}(p_B; k_0)$ or $\mathcal{S}^{\mathsf{L}}(k_0, p_B, c_0)$, L^2 is either $\mathsf{L_E}(p_A; k_0)$ or $\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1)$ and L^3 is $\mathsf{L}_{\$}(k_0)$; after that, from k_1 he is able to compute $c_1, ..., c_l$ with $c_i = y_i \oplus m_i^b$ (for $i = l$, $c_i = \pi_{|m_l^b|}(y_i) \oplus m_i^b$), $y_i = \mathsf{E}_{k_i}(p_B)$ and $k_i = \mathsf{E}_{k_{i-1}}(p_A)$ and the leakages $\mathsf{L}_E(p_B; k_1), \mathsf{L}_{\oplus}(m_1^b, y_1), \mathsf{L_E}(p_A; k_1), ..., \mathsf{L_E}(p_A; k_{l-1}), \mathsf{L}_E(p_B; k_l)$, $\mathsf{L}_{\oplus}(m_l^b, y_l)$; then, he computes $h = \mathsf{H}(c_0 \| ... \| c_{l-1} \| c_l)$ and he runs his oracle on input h to obtain c_{l+1} with the leakage L^4 which may be either $\mathsf{L}_{\mathsf{F}^*}(k_0, h; k)$ or $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k, k_0, h, c_{l+1})$. Finally, he answers $c = (c_0, ..., c_l, c_{l+1})$ with the leakages $(\mathsf{L}^1, \mathsf{L}^2, \mathsf{L}^3, \mathsf{L}_E(p_B; k_1), \mathsf{L}_{\oplus}(m_1^b, y_1), \mathsf{L_E}(p_A; k_1), ..., \mathsf{L_E}$ $\mathsf{L}_E(p_B; k_l)$, $\mathsf{L}_{\oplus}(m_l^b, y_l), \mathsf{L}^4)$. To answer to A^{L} uses one query to his oracle and time bounded by

$$t_{\$} + 2lt_{\mathsf{E}} + 2lt_{\mathsf{L(E)}} + lt_{\mathsf{L}(\oplus)} + t_{\mathsf{H}} \leq$$

$$t_{\$} + 2Lt_{\mathsf{E}} + 2Lt_{\mathsf{L(E)}} + Lt_{\mathsf{L}(\oplus)} + t_{\mathsf{H}} = t'$$

When A^{L} outputs a bit b', B^{L} outputs 1 if $b = b'$ (that is, A^{L} has won the $\mathsf{EavL2}$ game; otherwise, 0.
B^{L} does one query to his oracle and runs in time bounded by $t + t'$.
We observe that, if the oracle B^{L} faces, is implemented with PSVsfb, B^{L} correctly simulates Game 0 for A^{L}; otherwise, Game 1.

Bounding $|\Pr[E_0] - \Pr[E_1]|$ We can observe that $\Pr[E_0] = \Pr[\mathsf{B}^{\mathsf{L}}(\mathsf{PSVsfb}) \Rightarrow 1]$ and $\Pr[E_1] = \Pr[\mathsf{B}^{\mathsf{L}}(\mathsf{PSVsfb}^I) \Rightarrow 1]$. Since B^{L} is a $(q_{\mathsf{L}}, t + t')$-adversary, F^* is a $(1, t + t' + t_2, \epsilon_{\mathsf{STPRP}})$-$\mathsf{STPRP}$, whose implementation has $(1, q_{\mathcal{S}'}, t_{\mathcal{S}'}, t + t' + t_1)$- indistinguishable leakage, let E be a $(2, t + t' + t_3, \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$, whose implementation has $(q_{\mathcal{S}}, q'_{\mathcal{S}}, q_{\mathsf{L}}, t_{\mathcal{S}}, t'_{\mathcal{S}}, t + t' + t_3, \epsilon_{\mathsf{2\text{-}sim'}})$-2-simulatable' leakage,

we can use Lemma 4. Thus:

$$|\Pr[E_0] - \Pr[E_1]| = |\Pr[\mathsf{B}^{\mathsf{L}}(\mathsf{PSVsfb}) \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{L}}(\mathsf{PSVsfb}^I) \Rightarrow 1]| \leq$$

$$\epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{2\text{-}sim'}} + \epsilon_{\mathsf{PRF}}$$

Game 2 It is Game 1, where we have replaced $y_1 = \mathsf{E}_{k_1}(p_B), ..., y_l = \mathsf{E}_{k_l}(p_B)$ and $k_2, ..., k_l$ with random value. Moreover, the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_1)$, $\mathsf{L}_{\mathsf{E}}(p_A; k_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l-1})$, $\mathsf{L}_{\mathsf{E}}(p_B; k_l)$ have been replaced by the simulated one $\mathcal{S}^{\mathsf{L}}(k_1, p_B, y_1)$, $\mathcal{S}^{\mathsf{L}}(k_1, p_A, k_2)$,..., $\mathcal{S}^{\mathsf{L}}(k_{l-1}, p_A, k_l)$, $\mathcal{S}^{\mathsf{L}}(k_l, p_B, y_l)$. (Substantially we have replaced PSV with PSV^I). Let E_2 be the event that A^{L} guesses correctly the bit b.

Transition between Game 1 and 2 Although we cannot apply straightforwardly Lemma 2, we can reapply that proof, adding that every adversary, used in that proof, has to pick a random key k, 2 random values c_0 and c_{l+1}, compute $h = \mathsf{H}(c_0\|...\|c_l)$, collect the leakages $\mathsf{L}_{\$}(k_0)$, $\mathcal{S}^{\mathsf{L}}(k_0, p_B, c_0)$ and $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k, k_0, h, c_{l+1})$ (k_0 and $\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1)$ are obtained by all adversaries used in the proof of Lemma 2). Thus, all adversaries need additional $t'' = 3t_{\$} + t_{\mathsf{L}(\$)} + t_{\mathsf{H}} + t_{\mathcal{S}} + t_{\mathcal{S}'}$.

Bounding $|\Pr[E_2] - \Pr[E_1]|$ Since $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \longmapsto \{0,1\}^n$ be a $(2, t + t'' + t_4 + t_5), \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ whose implementation has a leakage function L which has $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_{\mathsf{L}}, t + t'' + t_4 + t_6, \epsilon_{\mathsf{2\text{-}sim}})$-2-simulatable leakage and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator, we can apply the bound of Lemma 2. Thus,

$$|\Pr[E_2] - \Pr[E_1]| = L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}})$$

Computing $\Pr[E_2]$ To compute $\Pr[E_2]$, again, we cannot reapply straightforwardly Lemma 3, but we can reapply that proof adding that every adversary, used in that proof, has to pick a random key k, 2 random values c_0 and c_{l+1}, compute $h = \mathsf{H}(c_0\|...\|c_l)$, collect the leakages $\mathsf{L}_{\$}(k_0)$, $\mathcal{S}^{\mathsf{L}}(k_0, p_B, c_0)$ and $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k, k_0, h, c_{l+1})$ (k_0 and $\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1)$ are obtained by all adversaries used in the proof of Lemma 2). Thus, all adversaries need additional t''.
Thus, since $\mathsf{PSV}s^I$ is $(q_{\mathsf{L}}, t + t'' + t_7, \epsilon_{\mathsf{EavL2}})$-secure, then,

$$\Pr[E_3] \leq \frac{1}{2} + L\epsilon_{\mathsf{EavL2s}}.$$

Bonding $\Pr[E_0]$ Applying everything we obtain that:

$$\Pr[E_0] \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{2\text{-}sim'}} + \epsilon_{\mathsf{PRF}} + L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{PRF}}) + \frac{1}{2} + L\epsilon_{\mathsf{EavL2s}} \leq$$

$$\frac{1}{2} + \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{2\text{-}sim'}} + (L+1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{EavL2s}})$$

which concludes the proof

The CPAL2 security From the EavL2 security of CONCRETE we can derive the CPAL2 and CCAL2. First, we observe that the decryption of the challenge query does not give additional information via leakage, if we suppose that the leakage $\mathsf{L}_{\oplus}(x,y)$ gives the same information as $\mathsf{L}_{\oplus}(x,z)$ (with $z = x \oplus y$) and the leakage $\mathcal{S}_{\mathsf{L}_{\mathsf{F}^*}}(k,x,y,z)$ gives the same information as the leakage $\mathcal{S}_{\mathsf{L}_{\mathsf{F}^*,-1}}(k,z,y,x)$ (with $z = \mathsf{F}^*_k(x,y)$).

With the following theorem we reduce the CPAL2 security of CONCRETE to the EavL2s-security of PSVs^I:

Theorem 9. *Let* F^* *be a* $(q_E+1, t+t_2, \epsilon_{\mathsf{STPRP}})$-STPRP*, whose implementation has* $(q_E+1, q_{\mathcal{S}'}, t+t_1, t_{\mathcal{S}'})$*- indistinguishable leakage, let* E *be a* $(2, t+t_3+max(t_4+t_5, t_6+t_7+t_8), \epsilon_{\mathsf{PRF}})$-PRF*, whose implementation has* $(q_{\mathsf{L}}, q_{\mathcal{S}}, t+t_3+t_6+t_7+t_9, t_{\mathcal{S}}, \epsilon_{\text{2-sim}})$*-2-simulatable leakage and* $(q_{\mathsf{L}}, q_{\mathcal{S}}, q_{\mathcal{S}'}, t+t_3+t_4+t_5, t_{\mathcal{S}}+t_{\mathcal{S}'}, \epsilon_{\text{2-sim}'})$*-2-simulatable' leakage let* $\mathsf{PSV}s^I$ *be* $(q_{\mathsf{L}}, t+t_3+t_6+t_{10}, \epsilon_{\mathsf{EavL2s}})$-EavL2*-secure, then* CONCRETE*, if encrypts at most* L *block messages, is* (q_E, t, ϵ)-CPAL2*-secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \frac{q_E}{2^n} + \epsilon_{\text{2-sim}'} + (L+1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\text{2-sim}} + \epsilon_{\mathsf{EavL2s}})$$

where

$$t_1 = (q_E+1)\left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t_{\mathsf{F}^*}\right)$$

$$t_2 = (q_E+1)\left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}\right)$$

$$t_3 = q_E\left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}} +\right) + t_{\mathsf{f}(q_E)}$$

$$t_4 = t_{\$} + 2Lt_{\mathsf{E}} + 2Lt_{\mathsf{L}(\mathsf{E})} + Lt_{\mathsf{L}(\oplus)} + t_{\mathsf{H}}$$

$$t_5 = q_{\mathsf{L}}t_{\mathsf{L}(\mathsf{E})} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)}$$

$$t_6 = 3t_{\$} + t_{\mathsf{L}(\$)} + t_{\mathsf{H}} + t_{\mathcal{S}} + t_{\mathcal{S}'}$$

$$t_7 = (L-1)t_{\mathsf{L}(\oplus)} + \max_{i=1,\ldots,L-1}\left[(2i-1)t_{\$} + (2i-3)t_{\mathcal{S}} + (2(L-i)-1)(t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})})\right]$$

$$t_8 = q_{\mathsf{L}}t_{\mathsf{L}(\mathsf{E})} + 3t_{\mathcal{S}} + t_{\mathsf{L}(\oplus)} + 2t_{\$}$$

$$t_9 = q_{\mathsf{L}}t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{sim}} + t_{\mathsf{L}(\oplus)} + 2t_{\$}$$

$$t_{10} = (2L-5)t_{\$} + (2L-5)t_{\mathcal{S}} + (L-1)t_{\mathsf{L}(\oplus)}$$

where $t_{\$}$ *is the time needed to sample a random block,* $t_{\mathsf{L}(\$)}$ *is the time needed to collect the leakage of the random sampling of a block,* t_{E} *the needed to execute* E*,* $t_{\mathsf{L}(\mathsf{E})}$ *the time needed to collect the leakage of* E*,* $t_{\mathsf{L}(\oplus)}$ *the time needed to collect the leakage of the XOR* ($\oplus$) *of two blocks,* t_{H} *is the time needed to execute once the hash function,* t_{F^*} *is the time needed to execute once the* STPRP F^**,* t_{sim} *is the time needed to relay the content of the two* $\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}$ *and the* Gen-$\mathcal{S}$ *query and* $t_{\mathsf{f}(q_E)}$ *is the time needed to lazy sample the tweakable random permutation* q_E *times.*

Proof. We use, as usual, a sequence of games:
Game 0 It the standard $\mathsf{CPAL2}$ game. Let E_0 be the event that the adversary A wins the $\mathsf{CPAL2}$ game, that is, he guesses correctly the bit b.

Game 1 It is Game 0, where we have replaced the leakage $\mathsf{L}_{\mathsf{F}^*}$ with the simulated $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}$. Let E_1 be the event that the adversary guesses correctly the bit b.

Transition between Game 0 and 1 We build a $(q_E + 1, t + t_1)$-adversary B^{L} against the indistinguishability of the leakage of F^* based on A to bound the absolute difference $|\Pr[E_0] - \Pr[E_1]|$.

The $(q_E+1, t+t_1)$ **indistinguishability adversary** B^{L} B^{L} has to distinguish if the leakage he obtains is $\mathsf{L}_{\mathsf{F}^*}(k_0, h; k)$ or $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', k_0, h, d)$ for a random key k'. At the start of the game, B^{L} chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function H, which he gives to A^{L}. Moreover, B^{L} picks a key k uniform at random in $\{0,1\}^n$ and a bit b uniformly at random in $\{0,1\}$ which he keeps secret.
When A does a leaking encryption query on input $m^i = (m^i_1, ..., m^i_{l^i})$, B proceeds as follow: first, (1) he picks a random value r^i in $\{0,1\}^n$, he sets $k^i_0 = r^i$ and collects the leakage $\mathsf{L}_{\$}(k^*_0)$, moreover he checks if $l^i > L$, in this case he returns $\perp$, then, (2) from k_0 he computes all the values $c^i_0, ..., c^i_{l^i-1}, c^i_{l^i}$, where $c^i_j = y^i_j \oplus m^i_j$ with $y^i_j = \mathsf{E}_{k^i_j}(p_B)$ (for $j = 0$, $c^i_0 = y^i_0$; for $j = l^i$, $c^i_{l^i} = \pi_{|m^i_{l^i}|}(y_{l^i}) \oplus m^i_{l^i}$) and $k^i_{j+1} = \mathsf{E}_{k^i_j}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k^i_0), \mathsf{L}_{\mathsf{E}}(p_A; k^i_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_1), \mathsf{L}_{\oplus}(y^i_1, m^i_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k^i_{l^i-1}), \mathsf{L}_{\oplus}(y^i_{l^i+1}, m^i_{l^i})$, after that, (3) he computes $h^i = \mathsf{H}(c^i_0 \| ... \| c^i_{l^i-1} \| c^i_{l^i})$ and $c^i_{l^i+1} = \mathsf{F}^*_k(h^i, k^i_0)$. He calls his oracle receiving L^1, which is either the leakage $\mathsf{L}_{\mathsf{F}^*}(h^i, k^i_0; k)$ or the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k^i_0, c^i_{l^i})$; finally, (4) he answers to A c^i with $c^i = (c^i_0, ..., c^i_{l^i}, c^i_{l^i+1})$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_0), \mathsf{L}_{\mathsf{E}}(p_A; k^i_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_1), \mathsf{L}_{\oplus}(y^i_1, m^i_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k^i_{l^i-1}), \mathsf{L}_{\oplus}(y^i_{l^i+1}, m^i_{l^i}), \mathsf{L}^1)$.
To answer to a leaking encryption query B queries once his oracle and runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(l^i + 1)t_{\mathsf{E}} + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t_{\mathsf{F}^*} \leq$$

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L + 1)t_{\mathsf{E}} + 2(L + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t_{\mathsf{F}^*}.$$

When A does the challenge query on input $(m^{*,0}, m^{*,1})$ with $m^{*,j} = (m^{*,j}_1, ..., m^{*,j}_{l^{*,j}})$, with $j = 0, 1$, B proceeds as follow: first, he checks if $|m^{*,0}| = |m^{*,1}|$, if it is not the case he returns $\perp$, moreover he checks if $l^{*,b} > L$, if it is the case he answers $\perp$, after that, (1) he sets $m^*; = m^{*,b}$, he picks a random value r^* in $\{0,1\}^n$, he sets $k^*_0 = r^*$ and collects the leakage $\mathsf{L}_{\$}(k^*_0)$, then, (2) from k_0 he computes all the values $c^*_0, ..., c^*_{l^*-1}, c^*_{l^*}$, where $c^*_j = y^*_j \oplus m^*_j$ with $y^*_j = \mathsf{E}_{k^*_j}(p_B)$ (for $j = 0$, $c^*_0 = y^*_0$; for $j = l^*$, $c^*_{l^*} = \pi_{|m^*_{l^*}|}(y_{l^*}) \oplus m^*_{l^*}$) and $k^*_{j+1} = \mathsf{E}_{k^*_j}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k^*_0), \mathsf{L}_{\mathsf{E}}(p_A; k^*_0), \mathsf{L}_{\mathsf{E}}(p_B; k^*_1), \mathsf{L}_{\oplus}(y^*_1, m^*_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k^i_{l^*-1}), \mathsf{L}_{\oplus}(y^i_{l^*+1}, m^*_{l^*})$, after that, (3) he computes $h^* = \mathsf{H}(c^*_0 \| ... \| c^*_{l^*-1} \| c^*_{l^*})$

and $c^*_{l^*+1} = \mathsf{F}^*_k(h^*, k^*_0)$. He calls his oracle receiving L^1, which is either the leakage $\mathsf{L}_{\mathsf{F}^*}(h^*, k^*_0; k)$ or the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k^*_0, c^*_{l^*})$; finally, (4) he answers to A c^* with $c^* = (c^*_0, ..., c^*_{l^*}, c^*_{l^*+1})$ and the leakages $(\mathsf{L}_\$(k_0), \mathsf{L}_\mathsf{E}(p_B; k^*_0), \mathsf{L}_\mathsf{E}(p_A; k^*_0), \mathsf{L}_\mathsf{E}(p_B; k^*_1), \mathsf{L}_\oplus(y^*_1, m^*_1)$,..., $\mathsf{L}_\mathsf{E}(p_A; k^*_{l^*-1})$, $\mathsf{L}_\oplus(y^*_{l^*+1}, m^*_{l^*})$, $\mathsf{L}^1)$.
To answer to the challenge query B queries once his oracle and runs in time bounded by

$$t_\$ + t_{\mathsf{L}(\$)} + 2(L+1)t_\mathsf{E} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathsf{F}^*}.$$

At the end of the game A outputs a bit b', if $b = b'$, then, B outputs 1, otherwise 0.
B^L runs in time $t + t_1$ (since A runs in time t) and he does $q_E + 1$ query to his oracle, with

$$t_1 = (q_E + 1)\left(t_\$ + t_{\mathsf{L}(\$)} + 2(L+1)t_\mathsf{E} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathsf{F}^*}\right)$$

We observe that if the oracle B faces is implemented with $\mathsf{L}_{\mathsf{F}^*}(\cdot, \cdot; \cdot)$, B correctly simulates Game 0 for A; otherwise, Game 1.

Bounding $|\Pr[E_0] - \Pr[E_1]|$ Clearly $\Pr[\mathsf{B}^{\mathsf{L}_{\mathsf{F}^*}} \Rightarrow 1] = \Pr[E_0]$, while,
$\Pr[\mathsf{B}^{\mathcal{S}^{lf_{\mathsf{F}^*}}} \Rightarrow 1] = \Pr[E_1]$.
Since F^* has $(q_E + 1, q'_\mathcal{S}, t'_\mathcal{S}, t + t_1)$-indistinguishable leakage and B^L is a $(q_E + 1, t + t_1)$ adversary we have that

$$\left|\Pr[\mathsf{B}^{\mathsf{L}_{\mathsf{F}^*}} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathcal{S}^{lf_{\mathsf{F}^*}}} \Rightarrow 1]\right| = 0$$

thus $\Pr[E_0] = \Pr[E_1]$.

Game 2 It is Game 1, where we have replaced the STPRP F^* with the random permutation f^*. Let E_2 be the event that the adversary guesses correctly the bit b.

Transition between Game 1 and 2 We build a $(q_E + 1, t + t_2)$-adversary C against the STPRP F^* based on A to bound the absolute difference $|\Pr[E_1] - \Pr[E_2]|$.

The $(q_E + 1, t + t_2)$ STPRP **adversary** C C has to distinguish if he is interacting with an oracle implemented with the STPRP $\mathsf{F}^*_k(\cdot, \cdot)$ or with a tweakable random permutation f^* for a random key k. At the start of the game, C chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function H, which he gives to A^L. Moreover, C picks a key k' uniform at random in $\{0,1\}^n$ and a bit b uniformly at random in $\{0,1\}$ which he keeps secret.
When A does a leaking encryption query on input $m^i = (m^i_1, ..., m^i_{l^i})$, C proceeds as follow: first, (1) he picks a random value r^i in $\{0,1\}^n$, he sets $k^i_0 = r^i$ and collects the leakage $\mathsf{L}_\$(k^*_0)$, moreover he checks if $l^i > L$, in this case he returns $\perp$, then, (2) from k_0 he computes all the values $c^i_0, ..., c^i_{l^i-1}, c^i_{l^i}$,

where $c_j^i = y_j^i \oplus m_j^i$ with $y_j^i = \mathsf{E}_{k_j^i}(p_B)$ (for $j = 0$, $c_0^i = y_0^i$; for $j = l^i$, $c_{l^i}^i = \pi_{|m_{l^i}^i|}(y_{l^i}) \oplus m_{l^i}^i$) and $k_{j+1}^i = \mathsf{E}_{k_j^i}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, m_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, m_{l^i}^i)$, after that, (3) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i-1}^i \| c_{l^i}^i)$, he calls his oracle on input (h^i, k_0^i) receiving $c_{l^i+1}^i$ and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k_0^i, c_{l^i+1}^i)$; finally, (4) he answers to A c^i with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, m_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, m_{l^i}^i)$, $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k_0^i, c_{l^i}^i))$.
To answer to a leaking encryption query C queries once his oracle and runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(l^i + 1)t_{\mathsf{E}} + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}} \leq$$

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L + 1)t_{\mathsf{E}} + 2(L + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}.$$

When A does the challenge query on input $(m^{*,0}, m^{*,1})$ with $m^{*,j} = (m_1^{*,j}, ..., m_{l^{*,j}}^{*,j})$, with $j = 0, 1$, C proceeds as follow: first, he checks if $|m^{*,0}| = |m^{*,1}|$, if it is not the case he returns $\perp$, moreover he checks if $l^{*,b} > L$, if it is the case he answers $\perp$, after that, (1) he sets $m^* := m^{*,b}$, he picks a random value r^* in $\{0,1\}^n$, he sets $k_0^* = r^*$ and collects the leakage $\mathsf{L}_{\$}(k_0^*)$, then, (2) from k_0^i he computes all the values $c_0^*, ..., c_{l^*-1}^*, c_{l^*}^*$, where $c_j^* = y_j^* \oplus m_j^*$ with $y_j^* = \mathsf{E}_{k_j^*}(p_B)$ (for $j = 0$, $c_0^* = y_0^*$; for $j = l^*$, $c_{l^*}^* = \pi_{|m_{l^*}^*|}(y_{l^*}) \oplus m_{l^*}^*$) and $k_{j+1}^* = \mathsf{E}_{k_j^*}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0^*), \mathsf{L}_{\mathsf{E}}(p_A; k_0^*), \mathsf{L}_{\mathsf{E}}(p_B; k_1^*), \mathsf{L}_{\oplus}(y_1^*, m_1^*)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^*-1}^i), \mathsf{L}_{\oplus}(y_{l^*+1}^i, m_{l^*}^*)$, after that, (3) he computes $h^* = \mathsf{H}(c_0^* \| ... \| c_{l^*-1}^* \| c_{l^*}^*)$, he calls his oracle on input (h^*, k_0^*) and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k_0^*, c_{l^*}^*)$; finally, (4) he answers to A c^* with $c^* = (c_0^*, ..., c_{l^*}^*, c_{l^*+1}^*)$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k_0^*), \mathsf{L}_{\mathsf{E}}(p_A; k_0^*), \mathsf{L}_{\mathsf{E}}(p_B; k_1^*), \mathsf{L}_{\oplus}(y_1^*, m_1^*)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^*-1}^*), \mathsf{L}_{\oplus}(y_{l^*+1}^*, m_{l^*}^*)$, $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k_0^*, c_{l^*}^*))$.
To answer to the challenge query C queries once his oracle and runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L + 1)t_{\mathsf{E}} + 2(L + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}.$$

At the end of the game A outputs a bit b', if $b = b'$, then, C outputs 1, otherwise 0.
C runs in time $t + t_2$ (since A runs in time t) and he does $q_E + 1$ query to his oracle, with

$$t_2 = (q_E + 1)\left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L + 1)t_{\mathsf{E}} + 2(L + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}\right)$$

We observe that if the oracle B faces is implemented with F^*, C correctly simulates Game 1 for A; otherwise, Game 2.

Bounding $|\Pr[E_1] - \Pr[E_2]|$ Clearly $\Pr[\mathsf{C}^{\mathsf{F}_k^*(\cdot,\cdot)} \Rightarrow 1] = \Pr[E_1]$, while, $\Pr[\mathsf{C}^{\mathsf{f}^*(\cdot,\cdot)} \Rightarrow 1] = \Pr[E_2]$.

Since F^* is a $(q_E+1, t+t_2)$-STPRP and B^{L} is a $(q_E+1, t+t_2)$ adversary we have that

$$\left|\Pr[\mathsf{C}^{\mathsf{F}^*_k(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathsf{C}^{\mathsf{f}^*(\cdot,\cdot)} \Rightarrow 1]\right| \leq \epsilon_{\mathsf{STPRP}}$$

thus $|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{STPRP}}$.

The $(q_{\mathsf{L}}, t+t_3)$ $\mathsf{EavL2}$ **adversary** D Based on the $\mathsf{CPAL2}$ adversary, we build a $\mathsf{EavL2}$ adversary D.
At the start of the game D receives the two constants p_A and p_B and the hash function H which he forwards to A.
D is playing an $\mathsf{EavL2}$ against $\mathsf{CONCRETE}'$, which is $\mathsf{CONCRETE}$ where we have replaced the STPRP F^* with a random tweakable permutation g^* (since g^* is only used once in the game, to compute $c^*_{l^*+1}$, it is equivalent to suppose that $c^*_{l^*+1}$ is picked uniformly at random) and the leakage $\mathsf{L}_{\mathsf{F}^*}(\cdot,\cdot;\cdot)$ is replaced with the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(\cdot,\cdot,\cdot,\cdot)$.
When A does a leakage query, D does the same query.
When A does a leaking encryption query on input $m^i = (m^i_1, ..., m^i_{l^i})$, D proceeds as follow: first, (1) he picks a random value r^i in $\{0,1\}^n$, he sets $k^i_0 = r^i$ and collects the leakage $\mathsf{L}_{\$}(k^*_0)$, moreover he checks if $l^i > L$, in this case he returns $\perp$, then, (2) from k^i_0 he computes all the values $c^i_0, ..., c^i_{l^i-1}, c^i_{l^i}$, where $c^i_j = y^i_j \oplus m^i_j$ with $y^i_j = \mathsf{E}_{k^i_j}(p_B)$ (for $j=0$, $c^i_0 = y^i_0$; for $j = l^i$, $c^i_{l^i} = \pi_{|m^i_{l^i}|}(y_{l^i}) \oplus m^i_{l^i}$) and $k^i_{j+1} = \mathsf{E}_{k^i_j}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k^i_0), \mathsf{L}_{\mathsf{E}}(p_A; k^i_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_1), \mathsf{L}_{\oplus}(y^i_1, m^i_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k^i_{l^i-1}), \mathsf{L}_{\oplus}(y^i_{l^i+1}, m^i_{l^i})$, after that, (3) he computes $h^i = \mathsf{H}(c^i_0 \| ... \| c^i_{l^i-1} \| c^i_{l^i})$, he lazy samples f^*, obtaining $c^i_{l^i+1}$, and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k^i_0, c^i_{l^i+1})$; finally, (4) he answers to A c^i with $c^i = (c^i_0, ..., c^i_{l^i}, c^i_{l^i+1})$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_0), \mathsf{L}_{\mathsf{E}}(p_A; k^i_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_1), \mathsf{L}_{\oplus}(y^i_1, m^i_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k^i_{l^i-1})$, $\mathsf{L}_{\oplus}(y^i_{l^i+1}, m^i_{l^i})$, $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k^i_0, c^i_{l^i}))$.
To answer to a leaking encryption query D runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(l^i+1)t_{\mathsf{E}} + 2(l^i+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}} + t_{\mathsf{f}} \leq$$

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}} + t_{\mathsf{f}}$$

. When A does the challenge query on input $(m^{*,0}, m^{*,1})$ with $m^{*,j} = (m^{*,j}_1, ..., m^{*,j}_{l^{*,j}})$, with $j = 0, 1$, D forwards $(m^{*,0}, m^{*,1})$ as his challenge query. He receives the ciphertext $c^* = (c^*_0, ..., c^*_{l^*+1})$ and the leakage L^1, which he forwards to A.
At the end of the game A outputs a bit b', if $b = b'$, then, C outputs 1, otherwise 0.
C runs in time $t + t_3$ (since A runs in time t) and does only the challenge query to his oracle and q_{L} leakage query, with

$$t_3 = q_E \left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}} +\right) + t_{\mathsf{f}(q_E)}.$$

The event TPC We observe that, if the the following event *Tweakable Permutation Collision* (TPC)

$$TPC := \{\text{for } j = 1, ..., q_E \text{ s.t. } h^* = h^j,\ c^*_{l^*+1} \neq c^j_{l^j+1}\}$$

does not happen, the EavL2 adversary correctly simulates Game 2 for the CPAL2 adversary A. Event TPC happens when there is an incoherence in the way the STPRP is simulated. Since there are for c_{l^*+1}, at most q_E problematic values out of 2^n possible values and c_{l^*+1} is picked uniformly at random we can bound $\Pr[TPC] \leq \frac{q_E}{2^n}$.

Bounding $\Pr[E_2]$ Let E_3 the probability that D wins the EavL2 game. We observe that if D has correctly simulated Game 2 for A then D wins if and only if A wins. Thus $\Pr[E_2] \leq \Pr[E_3] + \Pr[TPC]$.
Since E is a $(2, t+t_3+max(t_4+t_5, t_6+t_7+t_8), \epsilon_{\mathsf{PRF}})$-PRF whose implementation has $(q_{\mathsf{L}}, q_{\mathcal{S}}, t+t_3+t_6+t_7+t_9, t_{\mathcal{S}}, \epsilon_{\text{2-sim}})$-2-simulatable leakage and $(q_{\mathsf{L}}, q_{\mathcal{S}}, q_{\mathcal{S}'}, t+t_3+t_4+t_5, t_{\mathcal{S}}+t_{\mathcal{S}'}, \epsilon_{twosim'})$-2-simulatable' leakage and PSVs^I is $(q_{\mathsf{L}}, t+t_3+t_6+t_10, \epsilon_{\mathsf{EavL2s}})$-EavL2-secure and D is a $(q_{\mathsf{L}}, t+t_3)$-EavL2-adversary, we can bound $\Pr[E_3]$ using Propo. 1

$$\Pr[E_3] \leq \frac{1}{2} + \epsilon_{\text{2-sim}'} + (L+1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\text{2-sim}} + \epsilon_{\mathsf{EavL2s}})$$

(when we apply the proof of Propo. 1, we can skip the first two games of Lemma 4 since we have already replaced the leakage $\mathsf{L}_{\mathsf{F}^*}(\cdot,\cdot;\cdot)$ of F^* with the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(\cdot,\cdot,\cdot,\cdot)$ and the STPRP F^* with the random tweakable permutation f^*).

Bounding $\Pr[E_0]$. Putting everything together we can bound

$$\Pr[E_0] \leq \epsilon_{\mathsf{STPRP}} + \frac{q_E}{2^n} + \frac{1}{2} + \epsilon_{\text{2-sim}'} + (L+1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\text{2-sim}} + \epsilon_{\mathsf{EavL2s}})$$

which concludes the proof.

CCAL2 security Since CONCRETE is CIML2-secure in the unbounded model and it is CPAL2-secure, we expect that it is CCAL2-secure. This is proved by the following theorem:

Theorem 10. *Let* F^* *be a* $(q_E+q_D+1, t+t_2, \epsilon_{\mathsf{STPRP}})$-STPRP*, whose implementation has* $(q_E+q_D+1, q_{\mathcal{S}'}, t+t_1, t_{\mathcal{S}'})$- *indistinguishable leakage, let* E *be a* $(2, t+max(t_3+t_4+t_5, t_3+t_6+t_7+t_8, t_{11}+t_{12}), \epsilon_{\mathsf{PRF}})$-PRF*, whose implementation has* $(q_{\mathsf{L}}, q_{\mathcal{S}}, t+t_3+t_6+t_7+t_9, t_{\mathcal{S}}, \epsilon_{\text{2-sim}})$*-2-simulatable leakage and* $(q_{\mathsf{L}}, q_{\mathcal{S}}, q_{\mathcal{S}'}, t+t_3+t_4+t_5, t_{\mathcal{S}}+t_{\mathcal{S}'}, \epsilon_{\text{2-sim}'})$*-2-simulatable' leakage, let* H *be a* $(0, t+t_{11}+t_{12}, \epsilon_{\mathsf{CR}})$*-collision resistant hash function, let* $\mathsf{PSV}s^I$ *be* $(q_{\mathsf{L}}, t+t_3+t_6+t_1 0, \epsilon_{\mathsf{EavL2s}})$-EavL2*-secure, then* CONCRETE*, if encrypts at most* L *block messages, is* (q_E, q_D, t, ϵ)-CCAL2*-secure with*

$$\epsilon \leq \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_E+q_D}{2^n} + \frac{q_D(L+1)(q_D+2q_E)}{2^{n+1}} +$$
$$(q_D+L+1)\epsilon_{\mathsf{PRF}} + \epsilon_{\text{2-sim}'} + L(\epsilon_{\text{2-sim}} + \epsilon_{\mathsf{EavL2s}})$$

where

$$t_1 = (q_E+q_D+1)\left(2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L(E)}} + t_{\mathsf{H}} + t_{\mathsf{F}^*}\right) + (q_E+1)\left(t_{\$} + t_{\mathsf{L}(\$)}\right)$$

$$t_2 = (q_E + q_D + 1)\left(2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L(E)}} + t_{\mathsf{H}} + t_{\mathcal{S}'}\right) + (q_E+1)\left(t_{\$} + t_{\mathsf{L}(\$)}\right)$$

$$t_3 = t_{\mathsf{f}(q_E+q_D)} + (q_E+q_D)(t_{\mathsf{H}}+t_{\mathcal{S}'}) + (t_{\mathsf{E}}+t_{\mathsf{L(E)}})(2(L+1)q_E+q_D) + q_E(t_{\$}+t_{\mathsf{L}(\$)})$$

$$t_4 = t_{\$} + 2Lt_{\mathsf{E}} + 2Lt_{\mathsf{L(E)}} + Lt_{\mathsf{L}(\oplus)} + t_{\mathsf{H}}$$

$$t_5 = q_{\mathsf{L}}t_{\mathsf{L(E)}} + 2t_{\mathcal{S}} + t'_{\mathcal{S}} + 3t_{\$} + t_{\mathsf{L}(\$)}$$

$$t_6 = 3t_{\$} + t_{\mathsf{L}(\$)} + t_{\mathsf{H}} + t_{\mathcal{S}} + t_{\mathcal{S}'}$$

$$t_7 = (L-1)t_{\mathsf{L}(\oplus)} + \max_{i=1,\ldots,L-1}\left[(2i-1)t_{\$} + (2i-3)t_{\mathcal{S}} + (2(L-i)-1)(t_{\mathsf{E}} + t_{\mathsf{L(E)}})\right]$$

$$t_8 = q_{\mathsf{L}}t_{\mathsf{L(E)}} + 3t_{\mathcal{S}} + t_{\mathsf{L}(\oplus)} + 2t_{\$}$$

$$t_9 = q_{\mathsf{L}}t_{\mathsf{L(E)}} + t_{\mathsf{sim}} + t_{\mathsf{L}(\oplus)} + 2t_{\$}$$

$$t_{10} = (2L-5)t_{\$} + (2L-5)t_{\mathcal{S}} + (L-1)t_{\mathsf{L}(\oplus)}$$

$$t_{11} = q_{\mathsf{L}}t_{\mathsf{L(E)}} + (q_E+q_D+1)(2(L+1)(t_{\mathsf{E}}+t_{\mathsf{L(E)}}) + t_{\mathsf{H}} + t'_{\mathcal{S}} + (q_E+1)(t_{\mathsf{L}(\$)} + t_{\$})$$

$$t_{12} = (q_E+q_D+2)(t_{\mathsf{H}} + 2(L+1)t_{\mathsf{E}}) + t_{\mathsf{f}(q_E+q_D+2)}$$

where $t_{\$}$ is the time needed to sample a random block, $t_{\mathsf{L}(\$)}$ is the time needed to collect the leakage of the random sampling of a block, t_{E} the needed to execute E, $t_{\mathsf{L(E)}}$ the time needed to collect the leakage of E, $t_{\mathsf{L}(\oplus)}$ the time needed to collect the leakage of the XOR ($\oplus$) of two blocks, t_{H} is the time needed to execute once the hash function, t_{F^*} is the time needed to execute once the STPRP F^**, t_{sim} is the time needed to relay the content of the two $\mathsf{E}, \mathsf{L}_{\mathsf{E}} \setminus \mathsf{E}, \mathcal{S}^{\mathsf{L}}$ and the Gen-$\mathcal{S}$ query and $t_{\mathsf{f}(q_E)}$ is the time needed to lazy sample the tweakable random permutation q_E times.*

Proof. We use, as usual, a sequence of games:
Game 0 It the standard CCAL2 game. Let E_0 be the event that the adversary A wins the CCAL2 game, that is, he guesses correctly the bit b.

Game 1 It is Game 0, where we have replaced the leakage $\mathsf{L}_{\mathsf{F}^*}$ with the simulated $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}$. Let E_1 be the event that the adversary guesses correctly the bit b.

Transition between Game 0 and 1 We build a $(q_E+q_D+1, t+t_1)$-adversary B^{L} against the indistinguishability of the leakage of F^* based on A to bound the absolute difference $|\Pr[E_0] - \Pr[E_1]|$.

The $(q_E+q_D+1, t+t_1)$ indistinguishability adversary B^{L} B^{L} has to distinguish if the leakage he obtains is $\mathsf{L}_{\mathsf{F}^*}(k_0, h; k)$ or $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', k_0, h, d)$ for a random key k'. At the start of the game, B^{L} chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function H, which he gives to A^{L}. Moreover, B^{L} picks a key k uniform at random in $\{0,1\}^n$ and a bit b uniformly at random in $\{0,1\}$ which he keeps secret.
When A does a leaking encryption query on input $m^i = (m^i_1, ..., m^i_{l^i})$, B proceeds as follow: first, (1) he picks a random value r^i in $\{0,1\}^n$, he sets $k^i_0 = r^i$ and collects the leakage $\mathsf{L}_{\$}(k^*_0)$, moreover he checks if $l^i > L$, in this case

he returns $\perp$, then, (2) from k_0^i he computes all the values $c_0^i, ..., c_{l^i-1}^i, c_{l^i}^i$, where $c_j^i = y_j^i \oplus m_j^i$ with $y_j^i = \mathsf{E}_{k_j^i}(p_B)$ (for $j = 0$, $c_0^i = y_0^i$; for $j = l^i$, $c_{l^i}^i = \pi_{|m_{l^i}^i|}(y_{l^i}) \oplus m_{l^i}^i$) and $k_{j+1}^i = \mathsf{E}_{k_j^i}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, m_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, m_{l^i}^i)$, after that, (3) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i-1}^i \| c_{l^i}^i)$ and $c_{l^i+1}^i = \mathsf{F}_k^*(h^i, k_0^i)$. He calls his oracle receiving L^1, which is either the leakage $\mathsf{L}_{\mathsf{F}^*}(h^i, k_0^i; k)$ or the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k_0^i, c_{l^i}^i)$; finally, (4) he answers to A c^i with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, m_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i)$, $\mathsf{L}_{\oplus}(y_{l^i+1}^i, m_{l^i}^i)$, $\mathsf{L}^1)$.
To answer to a leaking encryption query B queries once his oracle and runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(l^i + 1)t_{\mathsf{E})} + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t_{\mathsf{F}^*} \leq$$

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L + 1)t_{\mathsf{E})} + 2(L + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t_{\mathsf{F}^*}.$$

When A does a decryption query on input $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$, if $l^i > L$, B returns *perp*; otherwise first (1) he computes $h^i = \mathsf{H}(c_0^i \| ... | c_{l^i+1}^i)$, $k_0^i = \mathsf{F}^{*,-1}(h^i, c_{l^i+1}^i)$ and calls his oracle on input $(k, k_0^i, h^i, c_{l^i+1}^i)$ receiving L^i which is either the leakage $\mathsf{L}_{\mathsf{F}^{*,-1}}(c_{l^i+1}^i, h^i)$ or the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^{*,-1}}}(k', c_{l^i+1}^i, h^i, k_0^i)$, then, (2) from k_0^i he computes $\tilde{c_0^i} = \mathsf{E}_{k_0^i}(p_B)$, he collects the leakage $\mathsf{L}_{\mathsf{E}}(p_B; k_0^i)$ and verifies if $\tilde{c_0^i} = c_0^i$, if it is not the case B breaks the execution and skips to step (4); otherwise, after that, (3) he computes from k_0^i all the values $m_1^i, ..., m_{l^i-1}^i, m_{l^i}^i$, where $m_j^i = y_j^i \oplus c_j^i$ with $y_j^i = \mathsf{E}_{k_j^i}(p_B)$ (for $j = l^i$, $m_{l^i}^i = \pi_{|c_{l^i}^i|}(y_{l^i}) \oplus c_{l^i}^i$) and $k_{j+1}^i = \mathsf{E}_{k_j^i}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, c_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, c_{l^i}^i)$. Finally (4), B answers $(\perp, \mathsf{L}^i, \mathsf{L}_{\mathsf{E}}(p_B; k_0^i)$ if the ciphertext is deemed invalid; otherwise $(m_1^i, ..., m_{l^i}^i)$ as well with the leakage $(\mathsf{L}^i, \mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, c_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, c_{l^i}^i))$.
To answer to a leaking decryption query B queries once his oracle and runs in time bounded by

$$2(l^i + 1)t_{\mathsf{E})} + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t_{\mathsf{F}^*} \leq$$

$$2(L + 1)t_{\mathsf{E})} + 2(L + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t_{\mathsf{F}^*}.$$

When A does the challenge query on input $(m^{*,0}, m^{*,1})$ with $m^{*,j} = (m_1^{*,j}, ..., m_{l^{*,j}}^{*,j})$, with $j = 0, 1$, B proceeds as follow: first, he checks if $|m^{*,0}| = |m^{*,1}|$, if it is not the case he returns $\perp$, moreover he checks if $l^{*,b} > L$, if it is the case he answers $\perp$, after that, (1) he sets $m^* := m^{*,b}$, he picks a random value r^* in $\{0,1\}^n$, he sets $k_0^* = r^*$ and collects the leakage $\mathsf{L}_{\$}(k_0^*)$, then, (2) from k_0 he computes all the values $c_0^*, ..., c_{l^*-1}^*, c_{l^*}^*$, where $c_j^* = y_j^* \oplus m_j^*$ with $y_j^* = \mathsf{E}_{k_j^*}(p_B)$ (for $j = 0$, $c_0^* = y_0^*$; for $j = l^*$, $c_{l^*}^* = \pi_{|m_{l^*}^*|}(y_{l^*}) \oplus m_{l^*}^*$) and $k_{j+1}^* = \mathsf{E}_{k_j^*}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0^*), \mathsf{L}_{\mathsf{E}}(p_A; k_0^*), \mathsf{L}_{\mathsf{E}}(p_B; k_1^*), \mathsf{L}_{\oplus}(y_1^*, m_1^*)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^*-1}^i), \mathsf{L}_{\oplus}(y_{l^*+1}^i, m_{l^*}^*)$, after that, (3) he computes $h^* = \mathsf{H}(c_0^* \| ... \| c_{l^*-1}^* \| c_{l^*}^*)$

and $c^*_{l^*+1} = \mathsf{F}^*_k(h^*, k^*_0)$. He calls his oracle receiving L^1, which is either the leakage $\mathsf{L}_{\mathsf{F}^*}(h^*, k^*_0; k)$ or the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k^*_0, c^*_{l^*})$; finally, (4) he answers to A c^* with $c^* = (c^*_0, ..., c^*_{l^*}, c^*_{l^*+1})$ and the leakages $(\mathsf{L}_\$(k_0), \mathsf{L}_\mathsf{E}(p_B; k^*_0), \mathsf{L}_\mathsf{E}(p_A; k^*_0), \mathsf{L}_\mathsf{E}(p_B; k^*_1), \mathsf{L}_\oplus(y^*_1, m^*_1)$,..., $\mathsf{L}_\mathsf{E}(p_A; k^*_{l^*-1})$, $\mathsf{L}_\oplus(y^*_{l^*+1}, m^*_{l^*})$, $\mathsf{L}^1)$.
To answer to the challenge query B queries once his oracle and runs in time bounded by

$$t_\$ + t_{\mathsf{L}(\$)} + 2(L+1)t_\mathsf{E} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathsf{F}^*}.$$

At the end of the game A outputs a bit b', if $b = b'$, then, B outputs 1, otherwise 0.
B^L runs in time $t + t_1$ (since A runs in time t) and he does $q_E + 1$ query to his oracle, with

$$t_1 = (q_E + 1)\left(t_\$ + t_{\mathsf{L}(\$)} + 2(L+1)t_\mathsf{E} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathsf{F}^*}\right) +$$

$$q_D\left(2(L+1)t_{\mathsf{E})} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathsf{F}^*}\right) =$$

$$(q_E + q_D + 1)\left(2(L+1)t_\mathsf{E} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathsf{F}^*}\right) + (q_E + 1)\left(t_\$ + t_{\mathsf{L}(\$)}\right).$$

We observe that if the oracle B faces is implemented with $\mathsf{L}_{\mathsf{F}^*}(\cdot, \cdot; \cdot)$, B correctly simulates Game 0 for A; otherwise, Game 1.

Bounding $|\Pr[E_0] - \Pr[E_1]|$ Clearly $\Pr[\mathsf{B}^{\mathsf{L}_{\mathsf{F}^*}} \Rightarrow 1] = \Pr[E_0]$, while, $\Pr[\mathsf{B}^{\mathcal{S}^{lf_{\mathsf{F}^*}}} \Rightarrow 1] = \Pr[E_1]$.
Since F^* has $(q_E + q_D + 1, q'_\mathcal{S}, t'_\mathcal{S}, t + t_1)$-indistinguishable leakage and B^L is a $(q_E + 1, t + t_1)$ adversary we have that

$$\left|\Pr[\mathsf{B}^{\mathsf{L}_{\mathsf{F}^*}} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathcal{S}^{lf_{\mathsf{F}^*}}} \Rightarrow 1]\right| = 0$$

thus $\Pr[E_0] = \Pr[E_1]$.

Game 2 It is Game 1, where we have replaced the STPRP F^* with the random permutation f^*. Let E_2 be the event that the adversary guesses correctly the bit b.

Transition between Game 1 and 2 We build a $(q_E + 1, t + t_2)$-adversary C against the STPRP F^* based on A to bound the absolute difference $|\Pr[E_1] - \Pr[E_2]|$.

The $(q_E+1+q_D, t+t_2)$ STPRP **adversary** C C has to distinguish if he is interacting with an oracle implemented with the STPRP $\mathsf{F}^*_k(\cdot, \cdot)$ or with a tweakable random permutation f^* for a random key k. At the start of the game, C chooses two constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and an hash function H, which he gives to A^L. Moreover, C picks a key k' uniform at random in $\{0,1\}^n$ and a bit b uniformly at random in $\{0,1\}$ which he keeps secret.
When A does a leaking encryption query on input $m^i = (m^i_1, ..., m^i_{l^i})$, C proceeds as follow: first, (1) he picks a random value r^i in $\{0,1\}^n$, he sets $k^i_0 = r^i$

and collects the leakage $\mathsf{L}_{\$}(k_0^*)$, moreover he checks if $l^i > L$, in this case he returns $\perp$, then, (2) from k_0 he computes all the values $c_0^i, ..., c_{l^i-1}^i, c_{l^i}^i$, where $c_j^i = y_j^i \oplus m_j^i$ with $y_j^i = \mathsf{E}_{k_j^i}(p_B)$ (for $j = 0$, $c_0^i = y_0^i$; for $j = l^i$, $c_{l^i}^i = \pi_{|m_{l^i}^i|}(y_{l^i}) \oplus m_{l^i}^i$) and $k_{j+1}^i = \mathsf{E}_{k_j^i}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, m_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, m_{l^i}^i)$, after that, (3) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i-1}^i \| c_{l^i}^i)$, he calls his oracle on input $(+1, h^i, k_0^i)$ receiving $c_{l^i+1}^i$ and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k_0^i, c_{l^i+1}^i)$; finally, (4) he answers to A c^i with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, m_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, m_{l^i}^i)$, $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k_0^i, c_{l^i}^i))$.
To answer to a leaking encryption query C queries once his oracle and runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(l^i + 1)t_{\mathsf{E})} + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t'_{\mathcal{S}} \leq$$

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L + 1)t_{\mathsf{E})} + 2(L + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t'_{\mathcal{S}}.$$

When A does a decryption query on input $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$, if $l^i > L$, B returns *perp*; otherwise first (1) he computes $h^i = \mathsf{H}(c_0^i \| ... | c_{l^i+1}^i)$, calls his oracle on input $(-1, h^i, c_{l^i+1}^i)$ receiving k_0^i and simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*,-1}}(k', c_{l^i+1}^i, h^i, k_0^i)$, then, (2) from k_0^i he computes $\tilde{c_0^i} = \mathsf{E}_{k_0^i}(p_B)$, he collects the leakage $\mathsf{L}_{\mathsf{E}}(p_B; k_0^i)$ and verifies if $\tilde{c_0^i} = c_0^i$, if it is not the case B breaks the execution and skips to step (4); otherwise, after that, (3) he computes from k_0^i all the values $m_1^i, ..., m_{l^i-1}^i, m_{l^i}^i$, where $m_j^i = y_j^i \oplus c_j^i$ with $y_j^i = \mathsf{E}_{k_j^i}(p_B)$ (for $j = l^i$, $m_{l^i}^i = \pi_{|c_{l^i}^i|}(y_{l^i}) \oplus c_{l^i}^i$) and $k_{j+1}^i = \mathsf{E}_{k_j^i}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, c_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, c_{l^i}^i)$. Finally (4), C answers $(\perp, \mathsf{L}^i, \mathsf{L}_{\mathsf{E}}(p_B; k_0^i)$ if the ciphertext is deemed invalid; otherwise $(m_1^i, ..., m_{l^i}^i)$ as well with the leakage $(\mathsf{L}^i, \mathsf{L}_{\mathsf{E}}(p_B; k_0^i), \mathsf{L}_{\mathsf{E}}(p_A; k_0^i), \mathsf{L}_{\mathsf{E}}(p_B; k_1^i), \mathsf{L}_{\oplus}(y_1^i, c_1^i)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^i-1}^i), \mathsf{L}_{\oplus}(y_{l^i+1}^i, c_{l^i}^i))$.
To answer to a leaking decryption query B queries once his oracle and runs in time bounded by

$$2(l^i + 1)t_{\mathsf{E})} + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t_{\mathcal{S}'} \leq$$

$$2(L + 1)t_{\mathsf{E})} + 2(L + 1)t_{\mathsf{L}(\mathsf{E}} + t_{\mathsf{H}} + t_{\mathcal{S}'}.$$

When A does the challenge query on input $(m^{*,0}, m^{*,1})$ with $m^{*,j} = (m_1^{*,j}, ..., m_{l^{*,j}}^{*,j})$, with $j = 0, 1$, C proceeds as follow: first, he checks if $|m^{*,0}| = |m^{*,1}|$, if it is not the case he returns $\perp$, moreover he checks if $l^{*,b} > L$, if it is the case he answers $\perp$, after that, (1) he sets $m^*; = m^{*,b}$, he picks a random value r^* in $\{0,1\}^n$, he sets $k_0^* = r^*$ and collects the leakage $\mathsf{L}_{\$}(k_0^*)$, then, (2) from k_0^i he computes all the values $c_0^*, ..., c_{l^*-1}^*, c_{l^*}^*$, where $c_j^* = y_j^* \oplus m_j^*$ with $y_j^* = \mathsf{E}_{k_j^*}(p_B)$ (for $j = 0$, $c_0^* = y_0^*$; for $j = l^*$, $c_{l^*}^* = \pi_{|m_{l^*}^*|}(y_{l^*}) \oplus m_{l^*}^*$) and $k_{j+1}^* = \mathsf{E}_{k_j^*}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0^*), \mathsf{L}_{\mathsf{E}}(p_A; k_0^*), \mathsf{L}_{\mathsf{E}}(p_B; k_1^*), \mathsf{L}_{\oplus}(y_1^*, m_1^*)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^*-1}^i), \mathsf{L}_{\oplus}(y_{l^*+1}^i, m_{l^*}^*)$, after that, (3) he computes $h^* = \mathsf{H}(c_0^* \| ... \| c_{l^*-1}^* \| c_{l^*}^*)$,

he calls his oracle on input (h^*, k_0^*) and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k_0^*, c_{l^*}^*)$; finally, (4) he answers to A c^* with $c^* = (c_0^*, ..., c_{l^*}^*, c_{l^*+1}^*)$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k_0^*), \mathsf{L}_{\mathsf{E}}(p_A; k_0^*), \mathsf{L}_{\mathsf{E}}(p_B; k_1^*), \mathsf{L}_{\oplus}(y_1^*, m_1^*)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^*-1}^*), \mathsf{L}_{\oplus}(y_{l^*+1}^*, m_{l^*}^*)$, $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k_0^*, c_{l^*}^*))$.
To answer to the challenge query C queries once his oracle and runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}}) + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}.$$

At the end of the game A outputs a bit b', if $b = b'$, then, C outputs 1, otherwise 0.
C runs in time $t + t_2$ (since A runs in time t) and he does $q_E + 1 + q_D$ query to his oracle, with

$$t_2 = (q_E+1)\left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}\right) +$$

$$q_D\left(2(L+1)t_{\mathsf{E}}) + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t_{\mathcal{S}'}\right) =$$

$$(q_E+q_D+1)\left(2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t_{\mathcal{S}'}\right) + (q_E+1)\left(t_{\$} + t_{\mathsf{L}(\$)}\right).$$

We observe that if the oracle C faces is implemented with F^*, C correctly simulates Game 1 for A; otherwise, Game 2.

Bounding $|\Pr[E_1] - \Pr[E_2]|$ Clearly $\Pr[\mathsf{C}^{\mathsf{F}_k^*(\cdot,\cdot)} \Rightarrow 1] = \Pr[E_1]$, while, $\Pr[\mathsf{C}^{\mathsf{f}^*(\cdot,\cdot)} \Rightarrow 1] = \Pr[E_2]$.
Since F^* is a $(q_E+1, t+t_2)$-STPRP and B^{L} is a $(q_E+1+q_D, t+t_2)$ adversary we have that

$$\left|\Pr[\mathsf{C}^{\mathsf{F}_k^*(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathsf{C}^{\mathsf{f}^*(\cdot,\cdot)} \Rightarrow 1]\right| \leq \epsilon_{\mathsf{STPRP}}$$

thus $|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{STPRP}}$.

Game 3 It is Game 2 where we suppose that all decryption queries, which are not the decryption of the ciphertext provided as answer of a previous encryption query, are invalid. Let E_3 the probability that A guesses correctly the bit b.

Transition from Game 2 to Game 3 We build a $(q_E+1, q_D, t+t_{11})$-$\mathsf{CIML2}$ adversary D in the unbounded model to bound $\Pr[E_2] - \Pr[E_3]$.

The $(q_E+1, q_D, t+t_{11})$**-**$\mathsf{CIML2}$ **adversary** D At the start of the game D receives the two constants p_A and p_B and the hash function H which he forwards to A. Moreover, he picks uniformly at random a bit b in $\{0,1\}$ and a random key k' in $\{0,1\}^n$ which he keeps secret.
D is playing an $\mathsf{CIML2}$ against $\mathsf{CONCRETE}'$, which is $\mathsf{CONCRETE}$ where we have replaced the STPRP F^* with a random tweakable permutation f^*. Moreover, for A the leakage $\mathsf{L}_{\mathsf{F}^*}(\cdot,\cdot;\cdot)$ is replaced with the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(\cdot,\cdot,\cdot,\cdot)$.
When A does a leakage query, on input (u, w), D computes $\mathsf{L} = \mathsf{L}_E(u; w)$ and answers L to A. This takes time $t_{\mathsf{L}(\mathsf{E})}$.

When A does a leaking encryption query on input $m^i = (m_1^i, ..., m_{l^i}^i)$, first (1), D checks if $l^i > L$, in this case he returns $\perp$, otherwise, he picks a random value k_0^i and he does an encryption query on input (k_0^i, m^i) receiving $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ with the leakage k_0^i, then, (2) he collects the leakage $\mathsf{L}_\$(k_0^i)$ and from k_0^i he recomputes all the values $c_0^i, ..., c_{l^i-1}^i, c_{l^i}^i$, where $c_j^i = y_j^i \oplus m_j^i$ with $y_j^i = \mathsf{E}_{k_j^i}(p_B)$ (for $j = 0$, $c_0^i = y_0^i$; for $j = l^i$, $c_{l^i}^i = \pi_{|m_{l^i}^i|}(y_{l^i}) \oplus m_{l^i}^i$) and $k_{j+1}^i = \mathsf{E}_{k_j^i}(p_A)$; moreover, he collects the leakages $\mathsf{L}_\mathsf{E}(p_B; k_0^i), \mathsf{L}_\mathsf{E}(p_A; k_0^i), \mathsf{L}_\mathsf{E}(p_B; k_1^i), \mathsf{L}_\oplus(y_1^i, m_1^i)$,..., $\mathsf{L}_\mathsf{E}(p_A; k_{l^i-1}^i), \mathsf{L}_\oplus(y_{l^i+1}^i, m_{l^i}^i)$, after that, (3) he computes $h^i = \mathsf{H}(c_0^i \| ... \| c_{l^i-1}^i \| c_{l^i}^i)$ and he simulates the leakage $\mathcal{S}^{\mathsf{L}_\mathsf{F}*}(k', h^i, k_0^i, c_{l^i+1}^i)$; finally, (4) he answers to A c^i with $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ and the leakages $(\mathsf{L}_\$(k_0), \mathsf{L}_\mathsf{E}(p_B; k_0^i)$, $\mathsf{L}_\mathsf{E}(p_A; k_0^i)$, $\mathsf{L}_\mathsf{E}(p_B; k_1^i), \mathsf{L}_\oplus(y_1^i, m_1^i)$,..., $\mathsf{L}_\mathsf{E}(p_A; k_{l^i-1}^i), \mathsf{L}_\oplus(y_{l^i+1}^i, m_{l^i}^i), \mathcal{S}^{\mathsf{L}_\mathsf{F}*}(k', h^i, k_0^i, c_{l^i}^i))$. Moreover, he keeps in memory the query and its answer.
To answer to a leaking encryption query D needs time bounded by

$$t_\$ + t_{\mathsf{L}(\$)} + 2(l^i + 1)t_\mathsf{E}) + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathcal{S}'} \leq$$

$$t_\$ + t_{\mathsf{L}(\$)} + 2(L + 1)t_\mathsf{E}) + 2(L + 1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathcal{S}'}.$$

When A does a decryption query on input $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i$, if $l^i > L$, D returns $\perp$, otherwise first (1) he does a decryption query on input c^i receiving the decryption and the leakage $(k_0^i, \tilde{c_0^i})$, then (2) he computes $h^i = \mathsf{H}(c_0^i \| ... | c_{l^i+1}^i)$ and simulates the leakage $\mathcal{S}^{\mathsf{L}_\mathsf{F}*,-1}(k', c_{l^i+1}^i, h^i, k_0^i)$, after that, (3) from k_0^i he recomputes $\tilde{c_0^i} = \mathsf{E}_{k_0^i}(p_B)$, he collects the leakage $\mathsf{L}_\mathsf{E}(p_B; k_0^i)$ and verifies if $\tilde{c_0^i} = c_0^i$, if it is not the case B breaks the execution and skips to step (4); otherwise, after that, (3) he computes from k_0^i all the values $m_1^i, ..., m_{l^i-1}^i, m_{l^i}^i$, where $m_j^i = y_j^i \oplus c_j^i$ with $y_j^i = \mathsf{E}_{k_j^i}(p_B)$ (for $j = l^i$, $m_{l^i}^i = \pi_{|c_{l^i}^i|}(y_{l^i}) \oplus c_{l^i}^i$) and $k_{j+1}^i = \mathsf{E}_{k_j^i}(p_A)$; moreover, he collects the leakages $\mathsf{L}_\mathsf{E}(p_A; k_0^i), \mathsf{L}_\mathsf{E}(p_B; k_1^i), \mathsf{L}_\oplus(y_1^i, c_1^i)$,..., $\mathsf{L}_\mathsf{E}(p_A; k_{l^i-1}^i), \mathsf{L}_\oplus(y_{l^i+1}^i, c_{l^i}^i)$. Finally (4), D answers $(\perp, \mathsf{L}^i, \mathsf{L}_\mathsf{E}(p_B; k_0^i)$ if the ciphertext is deemed invalid; otherwise $(m_1^i, ..., m_{l^i}^i)$ as well with the leakage $(\mathsf{L}^i, \mathsf{L}_\mathsf{E}(p_B; k_0^i), \mathsf{L}_\mathsf{E}(p_A; k_0^i), \mathsf{L}_\mathsf{E}(p_B; k_1^i), \mathsf{L}_\oplus(y_1^i, c_1^i)$,..., $\mathsf{L}_\mathsf{E}(p_A; k_{l^i-1}^i), \mathsf{L}_\oplus(y_{l^i+1}^i, c_{l^i}^i))$. Moreover, he keeps in memory this decryption query and its validity and if it is fresh (that is, if it is not the answer of a previous encryption, or challenge, query).
To answer to a leaking decryption query D queries once his oracle and needs time bounded by

$$2(l^i + 1)t_\mathsf{E} + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathcal{S}'} \leq$$

$$2(L + 1)t_\mathsf{E} + 2(L + 1)t_{\mathsf{L}(\mathsf{E})} + t_\mathsf{H} + t_{\mathcal{S}'}.$$

When A does the challenge query on input $(m^{*,0}, m^{*,1})$ with $m^{*,j} = (m_1^{*,j}, ..., m_{l^{*,j}}^{*,j})$, with $j = 0, 1$, D checks if $|m^{*,0}| = |m^{*,1}| \leq Ln$, if it is not the case he returns $\perp$ to A, otherwise he sets $m^* = m^{*,b}$ and proceeds as follow:
first (1),he picks a random value k_0^* and he does an encryption query on input (k_0^*, m^*), receiving $c^* = (c_0^*, ..., c_{l^*}^*, c_{l^*+1}^*)$ with the leakage k_0^*, then, (2) he collects the leakage $\mathsf{L}_\$(k_0^*)$ and from k_0^* he recomputes all the values $c_0^*, ..., c_{l^*-1}^*, c_{l^*}^*$,

where $c_j^* = y_j^* \oplus m_j^*$ with $y_j^* = \mathsf{E}_{k_j^*}(p_B)$ (for $j = 0$, $c_0^* = y_0^v$; for $j = l^*$, $c_{l^*}^* = \pi_{|m_{l^*}^*|}(y_{l^*}) \oplus m_{l^*}^*$) and $k_{j+1}^* = \mathsf{E}_{k_j^*}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k_0^*), \mathsf{L}_{\mathsf{E}}(p_A; k_0^*), \mathsf{L}_{\mathsf{E}}(p_B; k_1^*), \mathsf{L}_{\oplus}(y_1^*, m_1^*)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^*-1}^*), \mathsf{L}_{\oplus}(y_{l^*+1}^*, m_{l^*}^*)$, after that, (3) he computes $h^* = \mathsf{H}(c_0^* \| ... \| c_{l^*-1}^* \| c_{l^*}^*)$ and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k_0^*, c_{l^*+1}^*)$; finally, (4) he answers to A c^* with $c^* = (c_0^*, ..., c_{l^*}^*, c_{l^*+1}^*)$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k_0^*), \mathsf{L}_{\mathsf{E}}(p_A; k_0^*), \mathsf{L}_{\mathsf{E}}(p_B; k_1^*), \mathsf{L}_{\oplus}(y_1^*, m_1^*)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k_{l^*-1}^*)$, $\mathsf{L}_{\oplus}(y_{l^*+1}^*, m_{l^*}^*)$, $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^*, k_0^*, c_{l^*}^*))$. Moreover, he keeps in memory the query and its answer.
To answer to the challenge query D needs time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E})} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t_{\mathcal{S}'}.$$

At the end of the game A looks if one of the decryption query he has in memory, is fresh and valid. If it is the case, he outputs this query, otherwise, he outputs the first decryption query he has done.
D runs in time $t + t_3$ (since A runs in time t) and does $q_E + 1$ encryption queries and q_D decryption queries, with

$$t_{11} = q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + (q_E + 1)\left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}\right) +$$

$$q_D(2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t_{\mathcal{S}'}) =$$

$$q_{\mathsf{L}} t_{\mathsf{L}(\mathsf{E})} + (q_E + q_D + 1)(2(L+1)(t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})}) + t_{\mathsf{H}} + t'_{\mathcal{S}} + (q_E + 1)(t_{\$} + t_{\mathsf{L}(\$)}).$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$ We observe that D has correctly simulated Game 2 for A if all fresh decryption queries made by A are invalid. Only if it is not the case, D may win (he do not always win since the freshness of a valid decryption query may be invalidated by a subsequent encryption query). Thus, $|\Pr[E_2] - \Pr[E_3]| \leq \Pr[\mathsf{D} \text{ wins}]$.
Since E is a $(2, t + t_{11} + t_{12}, \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ and H is a $(0, t + t_{11} + t_{12}, \epsilon_{\mathsf{CR}})$-collision resistant hash function, we can apply the Thm. 6 obtaining that

$$\Pr[\mathsf{D} \text{ wins}] \leq \epsilon_{\mathsf{CR}} + \frac{(q_D+1)(L+1)(q_D+2q_E+1)}{2^{n+1}} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}}$$

(since we have already replaced the STPRP F^* with a random tweakable permutation we can omit the term $\epsilon_{\mathsf{STPRP}}$ since we can skip the first transition in the proof of Thm. 6).

Improving the bound $|\Pr[E_2] - \Pr[E_3]|$ Since the decryption query the $\mathsf{CIML2}$ adversary D output at the end of the game, its validity has already been established. Thus, considering the proof of theorem 6, where we have studied the possibility that each decryption query is valid one by one, we can improve the previous bound:

$$\Pr[\mathsf{D} \text{ wins}] \leq \epsilon_{\mathsf{CR}} + \frac{q_D(L+1)(q_D+2q_E)}{2^{n+1}} + \frac{q_D}{2^n} + q_D\epsilon_{\mathsf{PRF}}$$

(it is enough in the proof of Thm. 6, when we bound $|\Pr[\mathsf{A} \text{ wins Game 3}] - \Pr[\mathsf{A} \text{ wins Game 2}]|$ we can assume that

$$\Pr[\mathsf{A} \text{ wins Game } 2^{q_D+1}] = \Pr[\mathsf{A} \text{ wins Game } 2^{q_D}] = 0$$

since, if the first q_D decryption queries D does are not valid, then, also the last is not valid since it is the repetition of one of the others q_D decryption queries.

The $(q_{\mathsf{L}}, t + t_3)$ $\mathsf{EavL2}$ **adversary** EE Based on the $\mathsf{CCAL2}$ adversary A, we build a $\mathsf{EavL2}$ adversary EE.
At the start of the game EE receives the two constants p_A and p_B and the hash function H which he forwards to A.
EE is playing an $\mathsf{EavL2}$ against $\mathsf{CONCRETE}'$, which is $\mathsf{CONCRETE}$ where we have replaced the STPRP F^* with a random tweakable permutation g^* (since g^* is only used once in the game, to compute $c^*_{l^*+1}$, it is equivalent to suppose that $c^*_{l^*+1}$ is picked uniformly at random) and the leakage $\mathsf{L}_{\mathsf{F}^*}(\cdot,\cdot;\cdot)$ is replaced with the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(\cdot,\cdot,\cdot,\cdot)$.
When A does a leakage query, EE does the same query.
When A does a leaking encryption query on input $m^i = (m^i_1, ..., m^i_{l^i})$, EE proceeds as follow: first, (1) he picks a random value r^i in $\{0,1\}^n$, he sets $k^i_0 = r^i$ and collects the leakage $\mathsf{L}_{\$}(k^*_0)$, moreover he checks if $l^i > L$, in this case he returns $\perp$, then, (2) from k_0 he computes all the values $c^i_0, ..., c^i_{l^i-1}, c^i_{l^i}$, where $c^i_j = y^i_j \oplus m^i_j$ with $y^i_j = \mathsf{E}_{k^i_j}(p_B)$ (for $j = 0$, $c^i_0 = y^i_0$; for $j = l^i$, $c^i_{l^i} = \pi_{|m^i_{l^i}|}(y_{l^i}) \oplus m^i_{l^i}$) and $k^i_{j+1} = \mathsf{E}_{k^i_j}(p_A)$; moreover, he collects the leakages $\mathsf{L}_{\mathsf{E}}(p_B; k^i_0), \mathsf{L}_{\mathsf{E}}(p_A; k^i_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_1), \mathsf{L}_{\oplus}(y^i_1, m^i_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k^i_{l^i-1}), \mathsf{L}_{\oplus}(y^i_{l^i+1}, m^i_{l^i})$, after that, (3) he computes $h^i = \mathsf{H}(c^i_0 \| ... \| c^i_{l^i-1} \| c^i_{l^i})$, he lazy samples f^*, obtaining $c^i_{l^i+1}$, and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k^i_0, c^i_{l^i+1})$; finally, (4) he answers to A c^i with $c^i = (c^i_0, ..., c^i_{l^i}, c^i_{l^i+1})$ and the leakages $(\mathsf{L}_{\$}(k_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_0), \mathsf{L}_{\mathsf{E}}(p_A; k^i_0), \mathsf{L}_{\mathsf{E}}(p_B; k^i_1), \mathsf{L}_{\oplus}(y^i_1, m^i_1)$,..., $\mathsf{L}_{\mathsf{E}}(p_A; k^i_{l^i-1})$, $\mathsf{L}_{\oplus}(y^i_{l^i+1}, m^i_{l^i})$, $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(k', h^i, k^i_0, c^i_{l^i}))$.
To answer to a leaking encryption query EE runs in time bounded by

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(l^i + 1)t_{\mathsf{E}) } + 2(l^i + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}} + t_{\mathsf{f}} \leq$$

$$t_{\$} + t_{\mathsf{L}(\$)} + 2(L + 1)t_{\mathsf{E}) } + 2(L + 1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}} + t_{\mathsf{f}}.$$

When A does a leaking decryption query on input $c^i = (c^i_1, ..., c^i_{l^i}, c^i_{l^i+1})$, if $l^i > L$, D returns $\perp$, otherwise he proceeds as follow: first, (1) he computes $h^i = \mathsf{H}(c^i_0 \| ... | c^i_{l^i+1})$, he lazy samples $k^i_0 = \mathsf{f}^{*,-1}(h^i, c^i_{l^i+1})$ and he simulates the leakage $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^{*,-1}}}(k', c^i_{l^i+1}, h^i, k^i_0)$, after that, (2) from k^i_0 he computes $\tilde{c^i_0} = \mathsf{E}_{k^i_0}(p_B)$, he collects the leakage $\mathsf{L}_{\mathsf{E}}(p_B; k^i_0)$ and verifies if $\tilde{c^i_0} = c^i_0$, if it is not the case EE breaks the execution and skips to step (3); we observe that, by hypothesis, since we are in Game 3, thus, all decryption query are invalid. Finally (3), D answers $(\perp, \mathsf{L}^i, \mathsf{L}_{\mathsf{E}}(p_B; k^i_0)$ if the ciphertext is deemed invalid; To answer to a leaking decryption query D queries once his oracle and needs time bounded by

$$t_{\mathsf{H}} + t_{\mathsf{f}} + t_{\mathcal{S}} + t_{\mathsf{E}} + t_{\mathsf{LE}}.$$

When A does the challenge query on input $(m^{*,0}, m^{*,1})$ with $m^{*,j} = (m_1^{*,j}, ..., m_{l^*,j}^{*,j})$, with $j = 0, 1$, EE forwards $(m^{*,0}, m^{*,1})$ as his challenge query. He receives the ciphertext $c^* = (c_0^*, ..., c_{l^*+1}^*)$ and the leakage L^1, which he forwards to A.
At the end of the game A outputs a bit b', EE outputs the same bit b'. EE runs in time $t + t_3$ (since A runs in time t) and does only the challenge query to his oracle and q_{L} leakage query, with

$$t_3 = q_E \left(t_{\$} + t_{\mathsf{L}(\$)} + 2(L+1)t_{\mathsf{E}} + 2(L+1)t_{\mathsf{L}(\mathsf{E})} + t_{\mathsf{H}} + t'_{\mathcal{S}}\right) +$$

$$t_{\mathsf{f}(q_E+q_D)} + q_D(t_{\mathsf{H}} + t_{\mathcal{S}} + t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})}) =$$

$$t_{\mathsf{f}(q_E+q_D)} + (q_E + q_D)(t_{\mathsf{H}} + t_{\mathcal{S}'}) + (t_{\mathsf{E}} + t_{\mathsf{L}(\mathsf{E})})(2(L+1)q_E + q_D) + q_E(t_{\$} + t_{\mathsf{L}(\$)}).$$

The event TPC We observe that, if the the following event *Tweakable Permutation Collision* (TPC)

$$TPC := \{\text{for } j = 1, ..., q_E \text{ s.t. } h^* = h^j,\ c_{l^*+1}^* \neq c_{l^j+1}^j\}$$

does not happen, the $\mathsf{EavL2}$ adversary correctly simulates Game 2 for the $\mathsf{CPAL2}$ adversary A. Event TPC happens when there is an incoherence in the way the STPRP is simulated. Since there are for c_{l^*+1}, at most q_E problematic values out of 2^n possible values and c_{l^*+1} is picked uniformly at random we can bound $\Pr[TPC] \leq \frac{q_E}{2^n}$.

Bounding $\Pr[E_3]$ Let E_4 the probability that EE wins the $\mathsf{EavL2}$ game. We observe that if $\mathsf{A3}$ has correctly simulated Game 3 for A then, $\mathsf{A3}$ wins if and only if A wins. Thus $\Pr[E_3] \leq \Pr[E_4] + \Pr[TPC]$.
Since E is a $(2, t+t_3+max(t_4+t_5, t_6+t_7+t_8), \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ whose implementation has $(q_{\mathsf{L}}, q_{\mathcal{S}}, t+t_3+t_6+t_7+t_9, t_{\mathcal{S}}, \epsilon_{\mathsf{2\text{-}sim}})$-2-simulatable leakage and $(q_{\mathsf{L}}, q_{\mathcal{S}}, q_{\mathcal{S}'}, t+t_3+t_4+t_5, t_{\mathcal{S}}+t_{\mathcal{S}'}, \epsilon_{twosim'})$-2-simulatable' leakage and PSVs^I is $(q_{\mathsf{L}}, t+t_3+t_6+t_10, \epsilon_{\mathsf{EavL2s}})$-$\mathsf{EavL2}$-secure and D is a $(q_{\mathsf{L}}, t+t_3)$-$\mathsf{EavL2}$-adversary, we can bound $\Pr[E_4]$ using Propo. 1

$$\Pr[E_4] \leq \frac{1}{2} + \epsilon_{\mathsf{2\text{-}sim}'} + (L+1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{EavL2s}})$$

(when we apply the proof of Propo. 1, we can skip the first two games of Lemma 4 since we have already replaced the leakage $\mathsf{L}_{\mathsf{F}^*}(\cdot,\cdot;\cdot)$ of F^* with the simulated one $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}(\cdot,\cdot,\cdot,\cdot)$ and the STPRP F^* with the random tweakable permutation f^*).

Bounding $\Pr[E_0]$. Putting everything together we can bound

$$\Pr[E_0] \leq \epsilon_{\mathsf{STPRP}} + \frac{q_E}{2^n} + \epsilon_{\mathsf{CR}} + \frac{q_D(L+1)(q_D+2q_E)}{2^{n+1}} + \frac{q_D}{2^n} +$$

$$q_D\epsilon_{\mathsf{PRF}} + \frac{1}{2} + \epsilon_{\mathsf{2\text{-}sim}'} + (L+1)\epsilon_{\mathsf{PRF}} + L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{EavL2s}}) =$$

$$\frac{1}{2} + \epsilon_{\mathsf{STPRP}} + \epsilon_{\mathsf{CR}} + \frac{q_E+q_D}{2^n} + \frac{q_D(L+1)(q_D+2q_E)}{2^{n+1}} +$$

$$(q_D + L + 1)\epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{2\text{-}sim}'} + L(\epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{EavL2s}})$$

which concludes the proof.

E CONCRETE

The leakage-resilient ivAE-scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

Gen
- $k \leftarrow \mathcal{K}$
- $s \leftarrow \mathcal{S}$

Public parameters:
- $p_A, p_B \in \{0,1\}^n,\ p_A \neq p_B$
- $\mathsf{H} := \mathsf{H}_s$

$\mathsf{Enc}_k(r, m)$
- Parse $m = (m_1, ..., m_l)$ with $|m_1| = ... = |m_{l-1}| = n$ and $|m_l| \leq n$
- $r = k_0$
- $c_0 = \mathsf{E}_{k_0}(p_B)$
- $k_1 = \mathsf{E}_{k_0}(p_A)$
- For $i = 1, ..., l-1$
  - $c_i = \mathsf{E}_{k_i}(p_B) \oplus m_i$
  - $k_{i+1} = \mathsf{E}_{k_i}(p_A)$
- $y_l = \mathsf{E}_{k_l}(p_B)$
- $c_l = \pi_{|m_l|}(y_l) \oplus m_l$
- $h = \mathsf{H}(c_0 \| c_1 \| ... \| c_l)$
- $c_{l+1} = \mathsf{F}_k^{*,h}(k_0)$
- Return $c = (c_0, ..., c_l, c_{l+1})$

$\mathsf{Dec}_k(c)$
- Parse $c = (c_0, ..., c_{l+1})$ with $|c_0| = ... = |c_{l-1}| = |c_{l+1}| = N$ and $|c_l| \leq N$ // Common: Step 1
- $h = \mathsf{H} = (c_0 \| ... \| c_{l+1})$ // Common: Step 2
- $k_0 = \mathsf{F}_k^{*,-1,h}(c_{l+1})$ // Common: Step 3
- $\tilde{c}_0 = \mathsf{E}_{k_0}(p_B)$ // Verification: Step 1
- If $c_0 \neq \tilde{c}_0$ // Verification: Step 2
  - Return $\perp$ // Verification: Step 3
- $k_1 = \mathsf{E}_{k_0}(p_A)$
- For $i = 1, ..., l-1$
  - $m_i = \mathsf{E}_{k_i}(p_B) \oplus c_i$
  - $k_{i+1} = \mathsf{E}_{k_i}(p_A)$
- $y_l = \mathsf{E}_{k_l}(p_B)$
- $m_l = \pi_{|m_l|}(y_l) \oplus c_l$
- Return $m = (m_1, ..., m_l)$

Fig. 4. The leakage resilient ivAE-scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ - Full description. If AD are considered $h = \mathsf{H}'(a, m)$ replaces $h = \mathsf{H}(m)$ in both the Enc and Dec description as explained in App. B.2