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

Francesco Berti(✉), Olivier Pereira, and François-Xavier Standaert

ICTEAM/ELEN/Crypto Group, Université catholique de Louvain,
B-1348, Louvain-la-Neuve, Belgium
{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 [8,26] (e.g., it is the only one accepted in TLS 1.3 [23]). Although these schemes are deemed secure in the black box model (that is, when adversaries have access only to the inputs and

J. Buchmann et al. (Eds.): AFRICACRYPT 2019, LNCS 11627, pp. 229–249, 2019.
https://doi.org/10.1007/978-3-030-23696-0_12

outputs), they may not be secure when implemented, because their implementation secrets leak via side-channels. For example, the computation time, the power consumption, or the electromagnetic radiation of an implementation may reveal information about its manipulated secrets [1,27,29–31]. These attacks have a broad applicability, especially in contexts where cryptographic implementations can be under adversarial control (e.g., in IoT applications). As a result, various types of countermeasures have been introduced in the literature.

A first approach is to embed protections directly at the (hardware or software) implementation level. Examples include the addition of noise, masking or hiding: these solutions aim to reduce the information leaked by the implementation, but they are expensive and depend on technological assumptions, which may be hard to enforce [30]. For example, compared to an unprotected implementation, a good masking scheme typically implies factors of overheads ranging from tenths to hundreds, depending on the desired security level [19].

A complementary approach is to design schemes (typically, modes of operation) which are inherently more secure against side-channel attacks (for example, by manipulating the plaintext and the key as little as possible, and by using a key only a few number of times before changing it) [18]. This general idea, which we will denote as the leakage-resilient approach, can be instantiated in two main manners. One option is to rely on underlying primitives (like PRPs, PRFs and hash functions) that are all protected with the same level of security: we will denote this option as *uniform implementations*. Alternatively, one can rely on the very strong (and therefore expensive) protection of the implementations of a few blocks (we will call them leak free implementations for simplicity) and try to minimize their use. Such *leveled implementations* are expected to bring significant performance gains for implementations with high physical security levels as soon as the messages' length is beyond a few blocks [11,32].

We insist that despite no implementation is perfectly leak free in practice, we support this concept because (i) we may have reasonable instantiations of close to leak free components by using very high-order masking [24], (ii) this approach anyway indicates hardware designers where their efforts should be concentrated for security against side-channel attacks, and (iii) we may hope for more graceful degradations in the future, which is an interesting scope for further research.

We note also that although leakage affects the two goals of *leakage-resilient authenticated encryption* (LRAE) (i.e., confidentiality and authenticity), this paper is focused on authenticity (for privacy we reuse previous works [11,32]). Based on this premise, we aim to achieve CIML2, that is, *Ciphertext Integrity with coin Misuse and Leakage in 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 primitive (that is, all inputs and outputs of every primitive and all the keys used by non leak free components).

In particular, this implies that, to have CIML2 in this model, the correct authentication tag *cannot* be recomputed during decryption, because, otherwise, it would be leaked [11,13]. As a result, most standard AE modes do not achieve CIML2 in this model. By now, all modes achieving CIML2 use at least 2 calls to the leak free primitive per encryption or decryption query [10,13,20]. Therefore, if c_{LF} is the cost of a call to a leak free primitive which, in our case, is a tweakable block cipher, c_{wp} is the cost of a call to a weakly protected primitive which, in our case, is a block cipher with the same block size, the best cost to process an l-block message is $2c_{\mathsf{LF}} + 4lc_{wp}$ [10] (since all modes uses a hash function, to do a fair comparison, we suppose that it is always done according to the Hirose construction, which cost 2 calls per block [22]). Typically, c_{LF} will be within the range of tens or hundred times c_{wp}. For short messages, this cost is dominated by the cost of the leak free primitive. Our goal in this paper is to improve results on this front in order to extend the benefits of leveled implementations to shorter message lengths. For this purpose, we reduce the number of calls to the leak free block to one. We also design a mode that allows the Release of Unverified Plaintexts (RUP), which is a convenient feature for devices with little secure memory (see for example the discussions in [2,3,5]).

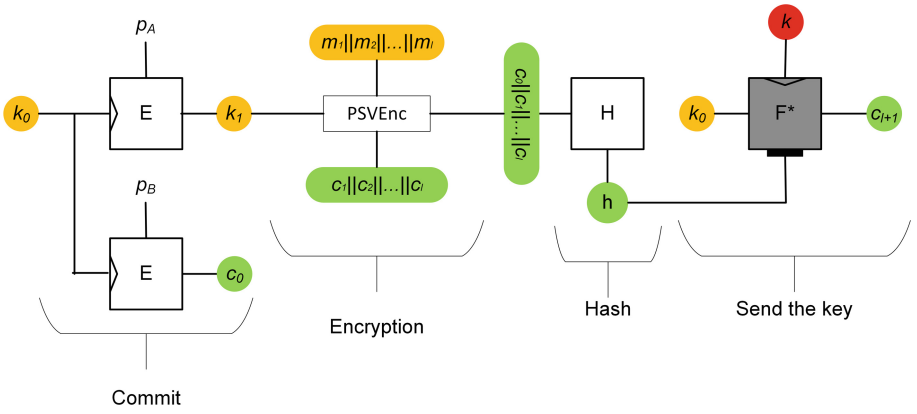

Fig. 1. The scheme CONCRETE *Commit* − *Encrypt* − *Send* − *the* − *Key*. We use red for long term secrets, orange for ephemeral ones and green for outputs. The gray shadowed primitive, F^*, is the leak free one. The key k_0 is randomly picked. It uses the PSV encryption scheme [32], described in Sect. 3. For decryption, first k_0 is recomputed $k_0 = \mathsf{F}_k^{*,-1}(h, c_{l+1})$, then c_0 is recomputed and checked. From k_0 decryption proceeds in the natural way. p_A and p_B are two n-bit constants, with $p_A \neq p_B$.

Our Contribution. We provide a new mode of operation which is CIML2-secure and uses only one leak free call per execution in both encryption and decryption: CONCRETE (for COmmit-eNCRypt-sEnd-The-kEy), see Fig. 1. The cost for l-block message is $c_{\mathsf{LF}} + 4(l+1)c_{wp}$. Thus, compared to [10], we approximately

halve the cost when the message is short and the leak free component is much more expensive than the weakly protected primitive. Previous modes proceed by deriving an ephemeral key from the long term key and a leak free component, and use that ephemeral key to encrypt the message. Our main idea is to avoid this key derivation step and to encrypt the message with a fresh random key k_0, obtaining $c_1, ..., c_l$. That key k_0 is then encrypted with the long term key, obtaining c_{l+1} so that it can be recovered by the receiver. To provide authenticity, first we put a commitment of k_0, c_0, then, we make c_{l+1} depend on all $c_1, ..., c_l$ and c_0. Thus, k_0 may be seen as the IV of the mode which is kept secret. The idea to put the IV secret is already present in [11], while Bellare [6] realises that encrypting the nonce (an IV which is not picked randomly, but only not repeated) improves the security of a scheme and prevents some protocol attacks. As the previous modes achieving CIML2 (for example [13]), the leak free primitive is a strong tweakable pseudorandom permutation (STPRP), which is inverted during the decryption. CONCRETE is the first AE scheme achieving CIML2 in the unbounded leakage model without using a range-oriented preimage-resistant hash function.

In addition, our mode is an AE scheme which is secure even when unverified plaintexts are released (i.e., it is a RUPAE) and it provides privacy in the presence of leakage (i.e., CPAL and CCAL [20]).

Related Works. Security definitions in the presence of leakage are given in the works of Barwell et al. [4] and Guo et al. [20]. Barwell et al. allow their primitives to leak in a limited manner (precisely, they exclude leakages from the challenge queries). The definitions of Guo et al. rather allow full leakage (including during the challenge queries) and nonce-resilience in the sense of Ashur et al. [3]. Guo et al. also use different leakage models (the unbounded model for authenticity and, for confidentiality, the simulatable one [33]), which is more amenable to leveled implementations. Therefore, we follow their definitional framework.

All previous modes achieving CIML2 in the unbounded model uses two calls to the leak free component: DTE2, EDT [13], FEMALE [20], TEDT [10] and TETSponge/S1P [21]. The last one is based on TEDT, achieves CIML2, but it uses a sponge, instead of PSV to encrypt. It is the basis for the NIST submission Spook [9].

Dobraunig et al. [16] and Bertoni et al. [15] propose two LRAE designs based on a sponge construction. Although their solutions are interesting and elegant, both lack a detailed analysis of the leakage-resilient properties (the recent works [17,21] do steps in this direction). 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] also propose an LRAE mode which supports strong composability results, but contrary to the previous ones, it is based on an uniform implementation, protecting everything in the same way. In practice, their concrete instances requires evaluating a pairing for each message block.

Structure of the Paper. We review in Sect. 2, the main definitions and notations used in the paper. Then, in Sect. 3 we present the specifications for our mode and the structure of previous constructions. After that, we present the rationale of the design of CONCRETE. The security statements of CIML2, AE, RUPAE, CPAL2 and CCAL2 security conclude the paper. Due to space constraints, only the sketchs of these proofs are provided.

In the extended version [14], the complete proofs can be found with all the details. Moreover, an extended background, a detailed analysis of previous works and the extension to associated data can be found.

2 Background

Notations. We use $(q_1, ..., q_d, t)$- bounded adversaries, who have access to the oracles $\mathcal{O}_1, ..., \mathcal{O}_d$, 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 strings, with $\{0,1\}^{\leq n}$ (resp. $\{0,1\}^*$) the set of all at most n bits long strings (resp. all finite strings) (that is, $\{0,1\}^{\leq n} = \bigcup_{i=1}^{n} \{0,1\}^i$ [resp. $\{0,1\}^* = \bigcup_{i=1}^{\infty} \{0,1\}^i$]). 0^n denotes the n zero string.

$x \xleftarrow{\$} \mathcal{X}$ denotes that the element x is picked uniformly at random from the set $\mathcal{X}$.

Given a probabilistic algorithm Alg, we call the support of $\mathsf{Alg}(x)$ the set $\mathsf{supp}(\mathsf{Alg}(x)) := \{y \in \{0,1\}^* \text{ s.t. } \Pr[\mathsf{Alg}(x) = y] > 0\}$, that is, the set of the possible outputs y from an input x.

A value is *fresh* if it has never appeared before in the history of the game.

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 (t, ϵ)-*collision resistant*: any t-bounded adversary has a probability at most ϵ to produce a collision, that is, given the key s, which is randomly picked, of the hash function, to output distinct $m_0, m_1 \in \mathcal{M}'$ s.t. $\mathsf{H}_s(m_0) = \mathsf{H}_s(m_1)$. From now on, 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_s as H. To simplify cost comparison, we assume that the hash is implemented with the Hirose construction [22], costing 2 block cipher calls per n-bit message block processed.

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 whether he is interacting with an $\mathsf{E}_k(\cdot)$ oracle, for a random key k, or with a random function $\mathsf{f}(\cdot)$ with the same signature as $\mathsf{E}_k(\cdot)$.

A (q, t, ϵ)- *strong tweakable pseudorandom permutation* (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 k or with a random tweakable permutation $\mathsf{f}(\cdot,\cdot)$ (that is, for every tw, $\mathsf{f}(tw,\cdot)$ is an independent random permuation) and its inverse $\mathsf{f}^{-1}(\cdot,\cdot)$, with the same signature as $\mathsf{F}_k(\cdot,\cdot)$ [28].

2.2 Authenticated Encryption (AE)

For authenticated encryption (AE) schemes we use the syntax introduced in Katz and Lindell [25] and also used in many works about AE, e.g., [7,26].

Definition 1. *An* authenticated encryption scheme AE *is a triple of algorithms* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *s.t. the keyspace* $\mathcal{K}$ *is a nonempty set, the encryption algorithm* Enc *is a probabilistic algorithm which takes as input the tuple* $(k, m) \in \mathcal{K} \times \mathcal{ME}$ *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* $\mathcal{ME}$ *or the symbol "*$\perp$*" (invalid).*

We require that the algorithms Enc *and* Dec *are the inverse of each other, asking the correctness* $(\forall m, k : m = \mathsf{Dec}_k(\mathsf{Enc}_k(m)))$ *and tidiness (if* $m = \mathsf{Dec}_k(c)$ *and* $m \neq \perp$ *then* $c \in \mathrm{supp}(\mathsf{Enc}_k(m)))$ *property. If* $m \leftarrow \mathsf{Dec}_k(c)$ *with* $m \neq \perp$ *we say that* c *is* valid*, otherwise it is* invalid*.*

We suppose that, to be probabilistic, Enc has internally access to a random source (for example, a (pseudo)random generator).

2.3 Security for Authenticated Encryption

The security guarantee that we expect from AE schemes is the following:

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 is bounded by* ϵ*:*

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

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 property provides both confidentiality and authenticity in the absence of leakages.

2.4 Ciphertext Integrity with Misuse and Leakage in Encryption and Decryption (CIML2)

Berti et al. [13] introduced the notion of ciphertext integrity with misuse and leakage in encryption and decryption (CIML2). Originally intended for nAE schemes, we adjust it to the syntax introduced in Definition 1.

A specific point here is that we require protection against corrupted sources of randomness used by the encryption oracle, hence offering a guarantee of randomness misuse resistance. To this purpose, in the definition below, we suppose that the adversary has control of the source of randomness used by Enc. In effect, in the game below, the adversary chooses and provides the randomness to the encryption oracle, which is now deterministic.

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 *(CIML2)-secure if, for every* (q_E, q_D, t)*-bounded adversary controlling the randomness used in the encryption process by the* EncL *oracle, we have*

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

(c^* fresh means that it was not previously returned by the EncL *oracle)*

Without leakage, it is mostly irrelevant if the adversary has or not has access to the decryption oracle [12,25]. This is not the case in the presence of leakage [11].

Leakage Model: The Unbounded Model. The definition above has leaking encryption and decryption oracles. We obviously need to define some limitations on what those leaking oracles leak.

We use a liberal model of leakages that seeks implementations with two levels of component protection. Our leakage model then distinguishes between:

- a strongly protected STPRP F^*, of which we make minimal use, and which we assume to not leak anything about its key, and
- a PRF and a hash function, which we assume to completely leak their internal state.

Following [11], we call this the *unbounded leakage model*.

2.5 Security When Unverified Plaintexts Are Released (RUPAE)

For this section, we follow [5] adapting their definitions to our AE syntax.

First, we observe that many decryption algorithms can be split in two: one part, SDec which decrypts, the other, SVer which verifies the authenticity. This is called *separated syntax* and a *separated* AE-scheme is $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{SDec}, \mathsf{SVer})$.

Now we can define the RUPAE security definition:

Definition 4 ([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*.*

A more detailed treatment can be found in [14].

Our focus in this paper is on authenticity. In the [14] of this document, we provide definitions of confidentiality with leakage (CPAL2 and CCAL2).

3 Design Specifications and Previous Solutions

Notations. For the leak free STPRP F^*, the PRF E and the hash function H, we assume $\mathcal{K} = \mathcal{M} = \mathcal{TW} = \mathcal{T} = \mathcal{B} = \{0,1\}^n$, $\mathcal{M}' = \{0,1\}^*$ and $\mathcal{ME} = \{0,1\}^{\leq Ln}$. A n-bit string is a *block*. Given a message m, we parse it in $(m_1, ..., m_l)$ with $|m_1| = ... = |m_{l-1}| = n$ and $1 < |m_l| \leq n$ (we sometimes also call m_l a block, regardless of its length).

The design goals of our proposed mode are as follows:

- AE secure in the black box model,
- CIML2 secure in the unbounded leakage model,
- CPAL and CCAL secure with some hypothesis about the leakage,
- only one call to the leak free component per execution,
- RUPAE (optional).

To reduce the possibility of leakage attacks via DPA (differential power analysis), the PRF E should not be used with more than two different plaintexts for any key. There is such an encryption mode, called PSV (see Sect. 3) [32] and some AE modes [10,11,13,20] (based on PSV which is based on the PRG proposed by Standaert et al. [33], which uses a PRF E and which is based on *rekeying*). The challenge of this PRG-based rekeying process lies in the choice of the first ephemeral key, on which we may repeatedly leak. Usually, LRAE modes 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), using a call to the leak free component;
2 Encryption, using PSV starting from k_1, using weakly/non protected components;
3 Authentication, again using a call to the leak free component.

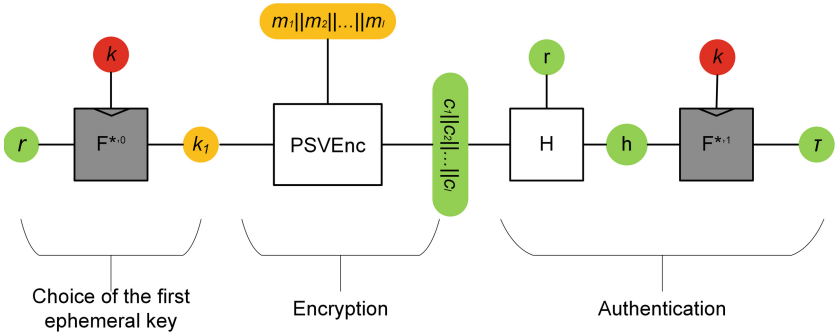

Fig. 2. The CIML2-secure scheme EDT [13] divided in the three parts.

Our goal in this paper is essentially to get rid of the leak free component used in the first part. An example of a construction divided in this three parts, can be found in Fig. 2.

PSV. [32] creates a (pseudo)random stream of block $y_1, ..., y_l$ which is XORed to $m_1, ..., m_l$. From an ephemeral key k_i, a new key $k_{i+1} = \mathsf{E}_{k_i}(p_A)$ and a new stream block $y_i = \mathsf{E}_{k_i}(p_B)$ are obtained. (p_A and p_B are two n bit strings with $p_A \neq p_B$)

A detailed analysis of various modes following this design pattern can be found in [14].

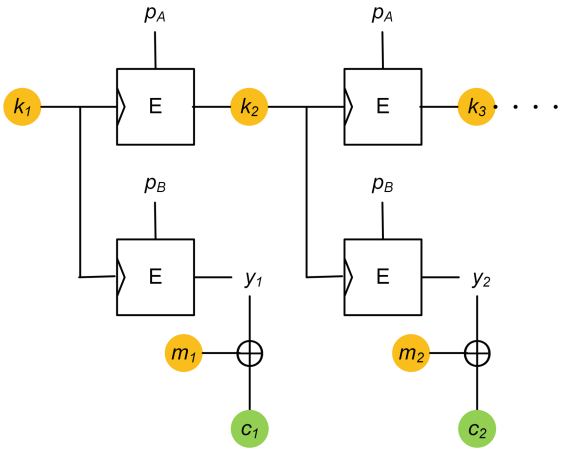

Fig. 3. The PSV encryption algorithm, presented at CCS 2015 by Pereira et al. [32].

4 Design Rationale of the Commit-Encrypt-Send-the-Key(CONCRETE)

We first describe CONCRETE (Fig. 1), then, we explain the ideas behind it. Finally we discuss some of its features.

CONCRETE (see Fig. 1) can be divided in the following steps:

- **Derivation of the first ephemeral key** We pick randomly k_0 as first ephemeral key. We use a round of the PRG of Standaert et al. [33] to obtain a commitment on k_0 (called c_0) and a fresh key (k_1). That is, using the public constants p_A and p_B (two n bit strings, 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 [32] (see Sect. 3) 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.
- **Sending the key** k_0 Since k_0 is picked uniformly at random, it must be recomputed by the decryption algorithm Dec from the ciphertext c. Thus, we send $c_{l+1} = \mathsf{F}^*_k(tw, k_0)$. Now, we need to choose tw.
- **Authenticity** To have it, we use for the tweak tw, the hash of the commitment c_0 and the output of the encryption part $c_1, ..., c_l$ obtaining $tw = h = \mathsf{H}(c_0 \| c_1 \| ... \| c_l)$ 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 $h = \mathsf{H}(c_0 \| ... \| c_l)$, then, k_0 is retrieved, with $k_0 = \mathsf{F}^{*,-1}_k(h, c_{l+1})$ and $\tilde{c_0} = \mathsf{E}_{k_0}(p_B)$ 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.

The main idea is avoiding to use a leak free component to generate the first ephemeral key k_1, but picking it uniformly at random, see Fig. 4a.

But this imposes to send the ephemeral key k_1 with the ciphertext, to allow the receiver to decrypt.

Next, we have the problem to send k_1. To do this we use the STPRP F^* to generate $c_{l+1} = \mathsf{F}^*_k(tw, k_1)$, because to send k_1 there is no other possibility than to use the master key k, see Fig. 4b. We have to decide what to put as tweak tw. Since we want to have authenticity, c_{l+1} must depend on all the other blocks. This can be done using $tw = h' = \mathsf{H}(c_1 \| ... \| c_l)$, see Fig. 4c. Unfortunately this solution gives no authenticity, since every ciphertext would be valid. It could be argued that such a scheme would be CCA-secure, since it could be proved that every decryption query made by an adversary, which is an not answer from a previous encryption query, would result in a random answer.

Thus, we add, in the ciphertext, a commitment c_0 of k_1, as $c_0 = \mathsf{E}_{k_1}(p_C)$ for a certain constant p_C (p_C must be different from p_B [and p_A], otherwise, the first block of plaintext would be leaked, or k_2). Thus, $h = \mathsf{H}(c_0 \| ... \| c_l)$ (see Fig. 4d). This scheme is CIML2 secure, when we recompute $\tilde{c_0}$ and check it.

But in this last scheme, k_1 is used three times with three different plaintexts as key of E. Thus, to avoid this, we pick randomly k_0, we compute its commit $c_0 = \mathsf{E}_{k_0}(p_B)$ and we do a rekeying to obtain $k_1 = \mathsf{E}_{k_0}(p_A)$ (see Fig. 4e).

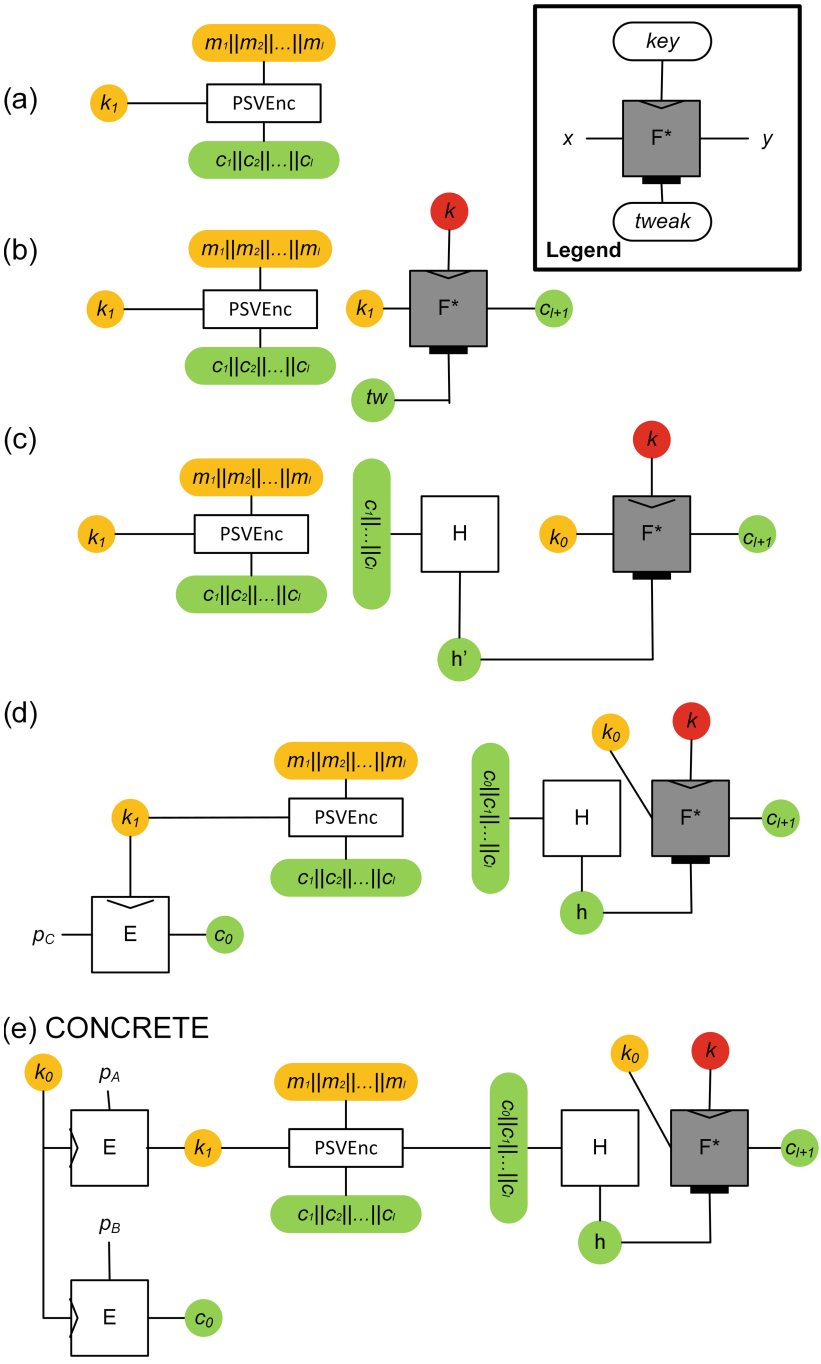

Fig. 4. How we designe CONCRETE.

CONCRETE has the following features:

- **Ciphertext expansion.** The ciphertext has an expansion of two blocks, that is, given $c \leftarrow \mathsf{Enc}_k(m)$, $|c| = |m| + 2n$.
- **Cost.** The cost of CONCRETE to process l block messages 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.

We note that our constructions could benefit from implementations of the leak-free component based on randomized countermeasures (such as masking, shuffling) in which case the PRG needed for both could be shared. Yet, this may not be systematic since the quality of the random numbers used in side-channel countermeasures may be weaker than for cryptographic keys.

A detailed description of the scheme can be found in [14], with, as well the extension to associated data.

5 Security Results for CONCRETE: CIML2, AE and RUPAE

For clarity, in this section and in the following one we do not consider the time bounds, which can be found in [14].

Notation. Given a ciphertext $c = (c_0, ..., c_l, c_{l+1})$ we define the *partial ciphertext* as the string $(c_0, ..., c_l)$, that is, the ciphertext without considering the block c_{l+1} encrypting the key.

5.1 CIML2 Security

Before proving the CIML2 security for CONCRETE, we define its leakage functions in the unbounded leakage model. We observe that $\mathsf{L}_E(r, m; k) := (k_0, m, c_{l+1})$, because, from them, all inputs, outputs and keys of the primitives, can be recomputed apart from the key k of the leak free F^*.

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 randomness misuse (when the adversary provides k_0), there is no useful information in the encryption leakage for CIML2 security.

Theorem 1. *Let* F^* *be a leak free* $(q_D + q_E + 1, \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudo-random permutation (*STPRP*), let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* ϵ_{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 in the unbounded leakage model with*

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

Observation on the Bound. We want to discuss some terms of the bound:

- $\epsilon_{\mathsf{STPRP}} + \frac{(q_E+q_D)(q_E+q_D-1)}{2^{n+1}}$ because F^* is a STPRP and not a PRF.
- ϵ_{CR} because, if there is a collision, the mode is trivially broken: given $c_0 = \mathsf{E}_{k_0}(p_B)$ if there is a collision $((c_0, c_1, ..., c_l), (c_0, c'_1, ..., c'_l)$ we observe that $c_{l+1} = c'_{l+1}$ if they both encrypt k_0,
- $(q_D + 1)\epsilon_{\mathsf{PRF}}$, because we do not check k_0, but $E_{k_0}(p_B)$.[1]
- $\frac{(q_D+1)(L+1)(q_D+2q_E)}{2^{n+1}}$ because we need that, in every decryption query, k_0 must have never been used before as ephemeral key; otherwise, c_0 would not be random anymore. 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, but this choice would require one more primitive 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 $\leq \epsilon_{\mathsf{PRF}}$.
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, $k_0 = \mathsf{F}^{*,-1}_k(h, c_{l+1})$ is still random. Then, again, the probability that $c_0 = \mathsf{E}_{k_0}(p_B)$ is $\leq \epsilon_{\mathsf{PRF}}$.

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 [14] 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 is based on the security of the PSV encryption scheme. We start studying confidentiality in the blackbox model:

Theorem 2. *Let* F^* *be a* $(q_D + q_E, \epsilon_{\mathsf{STPRP}})$*-strong tweakable pseudorandom permutation (*STPRP*), let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* H *be a* ϵ_{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)*-*AE *secure with*

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

[1] If we had checked k_0 and put it into the ciphertext, i.e., $c_0 = k_0$, we would have obtained a better CIML2 bound, but no AE security.

Observation on the Bound. In addition to the bound due to the $(q_E, q_D - 1)$-CIML2 security[2] (which is the same as for the ciphertext integrity) we have:

- $(q_E(L+1))\epsilon_{\mathsf{PRF}} + \frac{q_E(L+1)[q_E(L+1)-1]}{2^{n+1}}$ is due to PSV because we want that every ciphertext block is random,
  - in particular, $\frac{q_E(L+1)[q_E(L+1)-1]}{2^{n+1}}$ because we need that, in every encryption query, all keys used by E are different.

Proof (Sketch). First, we note that $c_0, ..., c_l$ can be seen as $(c_0, ..., c_l) = \mathsf{PSV}_{k_0}(0^n \| m)$. After that, we observe that the scheme is ciphertext-integrity secure (since it is 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 theorem with the time bounds (Thm. 7), its proof and a discussion of what happens if PSV is replaced with another scheme can be found in [14].

5.3 The RUPAE Security

Even if unverified plaintexts are released, CONCRETE remains 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* ϵ_{CR}*-collision resistant hash function. Then, the mode* CONCRETE*, which encrypts at most* L*-block long messages, is* (q_E, q_D, ϵ)*-*RUPAE *secure with* $Q = q_E + q_D$ *and* ϵ *bounded by:*

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

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

- $Q(L+1)\epsilon_{\mathsf{PRF}}$ due to PSV, since, if k_1 is random, PSV in decryption outputs a random string,
- $\frac{(L+1)Q[(L+1)Q-1]}{2^{n+1}}$, because we suppose that every ephemeral key used in an encryption or decryption query is different from all the others,
- $\epsilon_{\mathsf{STPRP}} + \frac{q_E(q_E-1)}{2^{n+1}}$ 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, it 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 theorem with the time bounds (Thm. 8), its proof and a discussion of what happens if PSV is replaced with another scheme can be found in [14].

[2] Observe that in the AE security definition, there is no more the final decryption query granted in the CIML2 security game.

6 Confidentiality with Leakage of CONCRETE

First, we introduce the leakage assumption we do on E: *simulatability*. Then, we discuss the security with leakage of PSV [32], in particular to what it is reduced to: the eavesdropper security with leakage (EavLDs) of an idealized single round variant of PSV called PSVs^I. Finally, we prove the CPAL2 and CCAL2 security of CONCRETE.

6.1 Leakage Model: Simulatability

For confidentiality it is necessary to bound the amount of information leaked by E, since $c_i = y_i \oplus m_i$ with $y_i = \mathsf{E}_{k_i}(p_B)$. To do this, we use the *simulatability* assumption: that is, let $y = \mathsf{E}_k(x)$, it is possible to create a simulator $\mathcal{S}^{\mathsf{L}}$, which has access only to x and y (not to k) and to the leakage function L. This simulator outputs a simulated leakage $\mathcal{S}(x, y, k')$ for a random k' which should be indistinguishable from the real one $\mathsf{L}_{\mathsf{E}}(x; k)$. This is captured by the following definition (Table 1):

Table 1. The q–sim experiment of Standaert et al. [33].

Game $\mathsf{q\text{–}sim}(\mathsf{A}, \mathsf{PRF}, \mathsf{L}, \mathcal{S}, b)$ [33, 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 $b = 1$
$\mathsf{E} \setminus \$(x)$	$\mathsf{E}_k(x)$, $\mathsf{L}(k, x)$	$\mathsf{E}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{E}_k(x))$
and one query of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}\text{–}\mathcal{S}(z, x)$	$\mathcal{S}^{\mathsf{L}}(z, x, k)$	$\mathcal{S}^{\mathsf{L}}(z, x, k^*)$

Definition 5 [q-simulatable leakages [33, Def. 1]]**.** *Let* E *be a PRF whose implementation has 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*

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

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 the implementation E.

Moreover, he has access to a special query, the $\mathsf{Gen}\text{–}\mathcal{S}$, because, since E is used in PSV, which is a scheme based on rekeying, the leakage of a previous round involves also the key used in the following round.

This assumption is useful to reduce the leakage security of the whole PSV encryption scheme to the leakage security of the encryption of a single block.

6.2 Other Leakage Assumptions

In addition, with respect to the PSV model [32], we need some additional hypothesis on the leakage of E and F^*. We start giving the reason why we need these additional hypothesis: k_0 has an additional source of leakage; $c_{l+1} := \mathsf{F}^*_k(h, k_0)$. Moreover, it is randomly picked and not output by E.

Thus, we need to simulate the leakage also of the STPRP $\mathsf{F}^*_k(\cdot,\cdot)$:

Definition 6. *The leak free implementation of the* STPRP $\mathsf{F}^*(\cdot,\cdot)$ *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^**.*

This hypothesis is given by the leak free assumption. Anyway, it is reasonable to believe that the adversary if he were able distinguish them, he would not be able to use this difference.

Thus, we are able to define the $\mathsf{q\text{–}sim}'$ experiment, which models the leakage of k_0:

Definition 7 [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 Definition 6). 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 $\mathsf{q\text{–}sim}'$ experiment is the $\mathsf{q\text{–}sim}$ with the following two modifications:

- first, the $\mathsf{Gen}\text{–}\mathcal{S}$ query is replaced by the $\mathsf{Gen}\text{–}\mathcal{S}'$ query which is answered by $\mathsf{L}_{\$}(k)$ if $b=0$; otherwise, by $\mathsf{L}_{\$}(k^*)$.
- second, the adversary is allowed to an additional $\mathsf{Key\text{–}Send}(h)$ query, which may be asked after having received the answer to the previous queries. If $b=0$, A receives $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(h,k,k^+,w)$; otherwise $\mathcal{S}^{\mathsf{L}_{\mathsf{F}^*}}_{\mathsf{F}^*}(h,k^*,k^+,w)$.

This models well the situation for k_0

6.3 The Eavesdropper Security with Leakage (EavLDs) Security of a Single Round Idealized Version of PSV

Similarly to what was done for PSV [32], we reduce the whole security of the scheme to the EavLDs security of an ideal version of PSVs^I, where the PRF $\mathsf{E}_{k_i^j}$

and its leakage are replaced with a random function and the simulated leakage, which encrypts only one block messages (see Table 2).

The EavLDs game (see Table 2) 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 $(q_{\mathsf{L}}, t, \epsilon)$-EavLD secure if the probability a (q_{L}, t)-adversary correctly guesses the bit b is bounded by $\frac{1}{2} + \epsilon$).

Table 2. The EavLDs experiment and the idealized single block version PSV^I. $\mathcal{S}$ is a simulator for the leakage of the PRF E. Note that k_1 is given as output for composability (see [14]).

<table>
<tr><td colspan="2">The EavLDs game and the idealized variant of PSV which encrypts a single block, PSVs^I</td></tr>
<tr><td>EavLDs
Initialization:
$k_0 \xleftarrow{\$} \mathcal{K},\ p_A, p_B \in \{0,1\}^n,\ p_A \neq p_B$
$b \xleftarrow{\$} \{0,1\}$

Challenge output:
$(m^{*,0}, m^{*,1}) \leftarrow \mathsf{A}^{\mathsf{L}}(p_A, p_B)$
$(c, k_1, \mathsf{L}_{\mathsf{PSVs}^I}) \leftarrow \mathsf{encsL}_{k_0}(m^{*,b})$

Finalization:
$b' \leftarrow \mathsf{A}^{\mathsf{L}}(c, k_1, \mathsf{L}_{\mathsf{PSVs}^I})$
If $|m_0| \neq |m_1| \vee |m_0| \neq n$, Return 0
If $b = b'$, Return 1, Else, Return 0</td><td>PSVs^I
Gen^I :
$k_0 \xleftarrow{\$} \{0,1\}^n$

$\mathsf{encs}^I_{k_0}(m)$:
$y \xleftarrow{\$} \{0,1\}^n$
$c = y \oplus m$
$k_1 \xleftarrow{\$} \{0,1\}^n$
Return (c, k_1)

$\mathsf{decs}^I_{k_0}(c)$ proceeds in the natural way.</td></tr>
<tr><td colspan="2">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 $k^- \xleftarrow{\$} \{0,1\}^n$.</td></tr>
</table>

For PSV the EavLD security (EavLDs for multiple block messages) is given by the following proposition:

Proposition 1 ([32]). *Let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$-PRF*, whose implementation has* $(q_{\mathsf{L}}, q_{\mathcal{S}}, \epsilon_{\mathsf{2\text{-}sim}})$*-2-simulatable leakage then,* PSV*, if it encrypts at most* L *block messages, is* ϵ-EavLD*-secure with* $\epsilon \leq L(\epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{2\text{-}sim}} + \epsilon_{\mathsf{EavLDs}})$.

6.4 CPAL2

Theorem 4. *Let* F^* *be a leak free* $(q_E + 1, \epsilon_{\mathsf{STPRP}})$-STPRP *whose implementation has* $(q_E + 1, q_{\mathcal{S}'})$*-indistinguishable leakage, let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$-PRF*, whose implementation has* $(q_{\mathsf{L}}, q_{\mathcal{S}}, \epsilon_{\mathsf{2\text{-}sim}})$*-2-simulatable leakage and*

$(q_{\mathsf{L}}, q_{\mathcal{S}}, q_{\mathcal{S}'}, \epsilon_{2\text{–sim}'})$-*2-simulatable leakage', let* $\mathsf{PSV}s^I$ *be* $(q_{\mathsf{L}}, \epsilon_{\mathsf{EavLDs}})$-$\mathsf{EavLDs}$-*secure, then,* $\mathsf{CONCRETE}$*, if it encrypts at most* L *block messages, is* (q_E, ϵ)-$\mathsf{CPAL2}$-*secure with*

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

About the bound we can observe:

– $L(\epsilon_{2\text{–sim}} + \epsilon_{\mathsf{EavLDs}} + \epsilon_{\mathsf{PRF}})$ is the EavLD-security of PSV [32].

The theorem with the time bounds (Thm. 9) and its proof can be found in [14].

Proof (Sketch). First, we reduce the EavLD security of $\mathsf{CONCRETE}$ to the EavLDs security of PSVs^I, using the same argument as Pereira et al. [32] (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 the 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 $\mathsf{CPAL2}$ adversary to an EavLD adversary.

6.5 CCAL2

Moreover, $\mathsf{CONCRETE}$ is $\mathsf{CCAL2}$ secure:

Theorem 5. *Let* F^* *be a leak free* $(q_E+q_D+1, \epsilon_{\mathsf{STPRP}})$-$\mathsf{STPRP}$ *whose implementation has* $(q_E+1, q_{\mathcal{S}'})$*-indistinguishable leakage, let* E *be a* $(2, \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$*, whose implementation has* $(q_{\mathsf{L}}, q_{\mathcal{S}}, \epsilon_{2\text{–sim}})$*-2-simulatable leakage and* $(q_{\mathsf{L}}, q_{\mathcal{S}}, q_{\mathcal{S}'}, \epsilon_{2\text{–sim}'})$*-2-simulatable leakage', let* H *be a* ϵ_{CR}*-collision resistant hash function, let* $\mathsf{PSV}s^I$ *be* $(q_{\mathsf{L}}, \epsilon_{\mathsf{EavLDs}})$-$\mathsf{EavLD}$-*secure, then,* $\mathsf{CONCRETE}$*, if encrypts at most* L *block messages, is* (q_E, q_D, ϵ)-$\mathsf{CCAL2}$-*secure with*

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

This bound is the $\mathsf{CIML2}$ bound + the $\mathsf{CPAL2}$ one ($\epsilon_{\mathsf{STPRP}}$ is not added twice because both proof shares the replacement of the STPRP F^* with a random tweakable permutation).

Proof (Sketch). We reuse the proof of the $\mathsf{CPAL2}$ security (Thm. 4). We add only that, due to the $\mathsf{CIML2}$-security in the unbounded model, the adversary can only ask 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 theorem with the time bounds (Thm. 10) and its complete proof can be found in [14].

7 Conclusion

With CONCRETE we have provided the first AE scheme achieving CIML2 in the unbounded model with the leak free is used only once. It provides also RUPAE and CPAL2 and CCAL2 [20]. This brings significant performance improvements, especially for short messages.

The leakage-resilience of this scheme crucially relies on the security of the leak free component. It would then be an interesting future challenge to investigate whether a weaker assumption could be made about this component: for instance, we may wonder whether an assumption of unpredictability could be sufficient.

Acknowledgments. François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union (EU) and the Walloon Region through the FEDER project USERMedia (convention number 501907-379156) and the ERC project SWORD (convention number 724725).

References

1. Albrecht, M.R., Paterson, K.G.: Lucky Microseconds: A Timing Attack on Amazon's *s2n* Implementation of TLS. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 622–643. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_24
2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 105–125. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_6
3. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 3–33. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_1
4. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated encryption in the face of protocol and side channel leakage. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 693–723. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_24
5. Barwell, G., Page, D., Stam, M.: Rogue decryption failures: reconciling AE robustness notions. In: Groth, J. (ed.) IMACC 2015. LNCS, vol. 9496, pp. 94–111. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27239-9_6
6. Bellare, M.: Symmetric ecryption revised. Technical report (2018). https://spotniq.files.wordpress.com/2018/07/spotniq18-se-revisited.pdf
7. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055718
8. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_41

9. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F.-X., Wiemer, F.: Spook: sponge-based leakage-resilient authenticated encryption with a masked tweakable block cipher (2019). https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/Spook-spec.pdf
10. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.-X.: TEDT, a leakage-resilient AEAD mode for high (physical) security applications. Cryptology ePrint Archive, Report 2019/137 (2019)
11. Berti, F., Koeune, F., Pereira, O., Peters, T., Standaert, F.-X.: Ciphertext integrity with misuse and leakage: definition and efficient constructions with symmetric primitives. In: AsiaCCS 2018, pp. 37–50 (2018)
12. Berti, F., Pereira, O., Peters, T.: Reconsidering generic composition: the tag-then-encrypt case. In: Chakraborty, D., Iwata, T. (eds.) INDOCRYPT 2018. LNCS, vol. 11356, pp. 70–90. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05378-9_4
13. Berti, F., Pereira, O., Peters, T., Standaert, F.-X.: On leakage-resilient authenticated encryption with decryption leakages. IACR Transactions on Symmetric Cryptology 2017(3), pp. 271–293 (2017)
14. Berti, F., Pereira, O., Standaert, F.-X.: Reducing the cost of authenticity with leakages: a CIML2-secure AE scheme with one call to a strongly protected tweakable block cipher. Cryptology ePrint Archive, Report 2019/451 (2019).https://eprint.iacr.org/2019/451
15. Bertoni, G., Daemen, J., Peters, M., Van Assche, G., Van Keer, R.: CAESAR submission: Ketje v2. Technical report (2016)
16. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: ISAP - towards side-channel secure authenticated encryption. Transactions on Symmetric Cryptology 2017(1), pp. 80–105 (2017)
17. Dobraunig, C., Mennink, B.: Leakage resilience of the duplex construction. IACR Cryptology ePrint Archive 2019, p. 225 (2019)
18. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS 2008, pp. 293–302 (2008)
19. Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 567–597. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_20
20. Guo, C., Pereira, O., Peters, T., Standaert, F.-X.: Leakage-resilient authenticated encryption with misuse in the leveled leakage setting: Definitions, separation results, and constructions. Cryptology ePrint Archive, Report 2018/484 (2018)
21. Guo, C., Pereira, O., Peters, T., Standaert, F.-X.: Towards lightweight side-channel security and the leakage-resilience of the duplex sponge (2019)
22. Hirose, S.: Some plausible constructions of double-block-length hash functions. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006). https://doi.org/10.1007/11799313_14
23. 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
24. Journault, A., Standaert, F.-X.: Very high order masking: efficient implementation and security evaluation. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 623–643. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_30
25. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, 2nd edn. CRC Press, Boca Raton (2014)

26. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44706-7_20
27. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
28. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_3
29. Longo, J., De Mulder, E., Page, D., Tunstall, M.: SoC it to EM: electromagnetic side-channel attacks on a complex system-on-chip. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 620–640. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_31
30. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Boston, MA (2007). https://doi.org/10.1007/978-0-387-38162-6
31. Mangard, S., Oswald, E., Standaert, F.-X.: One for all - all for one: unifying standard differential power analysis attacks. IET Inf. Secur. **5**(2), 100–110 (2011)
32. Pereira, O., Standaert, F.-X., Vivek, S.: Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In: ACM CCS 2015, pp. 96–108 (2015)
33. Standaert, F.-X., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 335–352. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_19