A preliminary version of this paper appears in the proceedings of the *11th International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2020)*. This is the full version.

Leakage-Resilient Authenticated Encryption from Leakage-Resilient Pseudorandom Functions

Juliane Krämer and Patrick Struck

Technische Universität Darmstadt, Germany
{jkraemer,pstruck}@cdc.tu-darmstadt.de

Abstract. In this work we study the leakage resilience of authenticated encryption schemes. We show that, if one settles for non-adaptive leakage, leakage-resilient authenticated encryption schemes can be built solely from leakage-resilient pseudorandom functions.
Degabriele et al. (ASIACRYPT 2019) introduce the FGHF′ construction which allows to build leakage-resilient authenticated encryption schemes from functions which, under leakage, retain both pseudorandomness and unpredictability. We revisit their construction and show the following. First, pseudorandomness and unpredictability do not imply one another in the leakage setting. Unfortunately, this entails that any instantiation of the FGHF′ construction indeed seems to require a function that is proven both pseudorandom and unpredictable under leakage. Second, however, we show that the unpredictability requirement is an artefact that stems from the underlying composition theorem of the N2 construction given by Barwell et al. (ASIACRYPT 2017). By recasting this composition theorem, we show that the unpredictability requirement is unnecessary for the FGHF′ construction. Thus, leakage-resilient AEAD schemes can be obtained by instantiating the FGHF′ construction with functions that are solely pseudorandom under leakage.

Keywords: AEAD · Leakage Resilience · Side Channels · FGHF′

1 Introduction

Authenticated encryption schemes with associated data (AEAD) are fundamental cryptographic primitives which enable Alice to send a ciphertext to Bob such that (1) Eve does not learn anything about the underlying message and (2) Bob can detect any manipulation of the ciphertext. In recent years, the study of AEAD schemes has received a lot of attention, for instance through the recent CAESAR competition [9] or the ongoing NIST standardization process on lightweight cryptography [25]. While AEAD schemes are well studied in the leak-free setting, their leakage resilience is not that well established, although several schemes [5, 12, 14, 15] which are designed to be secure in the presence of leakage have been proposed.

A notable work regarding leakage resilience of AEAD schemes is the work by Barwell et al. [5]. They show that the Encrypt-then-MAC paradigm [7] yields a leakage-resilient AEAD scheme if both the encryption scheme and the MAC are leakage-resilient. They also introduce the corresponding security notions. Recently, Degabriele et al. [14] refined this result by introducing the FGHF′ construction, showing that leakage-resilient encryption schemes and MACs can be build from fixed-input-length functions which are both pseudorandom and unpredictable under leakage. While leakage-resilient pseudorandomness is well established in the literature, leakage-resilient unpredictability has been defined by Degabriele et al. specifically for the FGHF′ construction. This security notion allows the adversary to obtain leakage for the input of which it predicts the output.[1] This raises the natural question:

What is the relation of pseudorandomness and unpredictability under leakage?

While pseudorandomness and unpredictability imply one another in the leak-free setting, Degabriele et al. claim that the notions are incomparable under leakage. We confirm their claim by providing two constructions, each being secure with respect to one notion while being insecure with respect to the other. This seems to entail that any instantiation of the FGHF′ construction indeed requires a function that is proven both pseudorandom and unpredictable under leakage. Given that leakage-resilient unpredictability is a new security notion, our separation result gives rise to another question:

Can leakage-resilient AEAD schemes be built solely from leakage-resilient pseudorandom functions?

[1] Note that the same does not work for pseudorandomness. Leakage of a single output bit allows to easily distinguish the function from a random function.

Surprisingly, we answer this question in the affirmative. We demonstrate that the necessity of leakage-resilient unpredictability stems from the composition theorem of Barwell et al. [5]. As observed in [14], this composition theorem imposes a security notion towards the MAC that prohibits constructing it from a leakage-resilient pseudorandom function. However, the composition theorem aims for arbitrary encryption schemes and MACs, while the encryption scheme and the MAC of the FGHF′ construction [14] exhibit a special structure. Thus, we show that recasting the composition theorem from [5] for these encryption schemes and MACs, allows to relax the security notion of the MAC such that it can be constructed from a leakage-resilient pseudorandom function. This comes at the cost of imposing a stronger security notion for the encryption scheme. However, it turns out that the encryption scheme underlying the FGHF′ construction — without any modification — achieves this stronger notion.

1.1 Our Contribution

Our contribution is threefold.

1) We show that, in contrast to the leak-free setting, pseudorandomness and unpredictability are not equivalent under leakage, thereby confirming a conjecture made in [14].

2) We recast the N2 composition theorem in the leakage setting by Barwell et al. [5], for a certain class of encryption schemes and MACs. We show that, in this case, other security notions for the encryption scheme and the MAC are sufficient to build leakage-resilient AEAD schemes. More precisely, we can weaken the security notion for the MAC at the cost of strengthening the security notion for the encryption scheme.

3) We revisit the FGHF′ construction [14] with respect to our recast composition theorem. We show that the encryption part (without any modification) achieves this stronger security notion. Regarding the MAC, we show that leakage-resilient pseudorandomness is sufficient to achieve the weaker security notion imposed by our recast composition theorem. This completely removes the necessity of leakage-resilient unpredictability to instantiate the FGHF′ construction, as opposed to the initial work [14]. Since proving leakage-resilient unpredictability turned out to be a main challenge for Slae [14] (a sponge-based instantiation of FGHF′), this is an important contribution towards building leakage-resilient AEAD schemes from simpler building blocks.

1.2 Related Work

The first leakage-resilient AEAD scheme RCB was proposed in [3] and broken shortly afterwards [2]. Another series of works propose leakage-resilient authenticated encryption schemes [10–12, 19, 21], symmetric encryption schemes and MACs [26]. In contrast to our setting, these works assume that only some components leak while the other components are assumed to be leak-free, for instance by using traditional countermeasures like masking [13]. Some of them also assume that the leakage is simulatable, an assumption that is not beyond dispute [22, 28]. Functions and permutations which are pseudorandom under leakage have been proposed for instance in [14, 16, 18, 29, 30]. Functions which are unpredictable under leakage have only been studied in [14] which also defined this notion.

1.3 Organization of the Paper

Section 2 provides the necessary background required for this work. In Section 3 we provide the motivation for our work by showing that, in the leakage setting, pseudorandomness and unpredictability of functions do not imply one another. We recast the composition theorem for the N2 construction by Barwell et al. [5] in Section 4. In Section 5, we show that the FGHF′ construction [14] achieves the security notions demanded by our recast composition theorem.

2 Preliminaries

2.1 Notation

We use the code-based game-playing framework by Bellare and Rogaway [8]. Each game consists of an initialize procedure, a finalize procedure, and several other procedures. The initialize procedure sets up the game and its output is given as input to the adversary which was oracle access to the other procedures, i.e., it can invoke them on the corresponding inputs and receives the output of the procedure in return. Eventually, the adversary will terminate generating some output which is given as input to the finalize procedure. The output of the finalize procedure determines whether the adversary has won the game or not. In this work we mainly use distinguishing games, in which the adversary has to determine a secret bit, i.e., the finalize procedure simply checks whether the adversary has guessed the correct bit or not. For an adversary $\mathcal{A}$ and a game G, we write $\mathsf{G}^{\mathcal{A}} \Rightarrow x$ to denote that the output of G, when interacting with $\mathcal{A}$, is x. Likewise, we write $\mathcal{A}^{\mathsf{G}} \Rightarrow x$ to denote that $\mathcal{A}$, when playing G, outputs x.

2.2 Primitives

An *authenticated encryption scheme with associated data* $\text{AEAD} = (\mathtt{E}, \mathtt{D})$ is a pair of efficient algorithms such that:

- The deterministic encryption algorithm $\mathtt{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \{0,1\}^*$ takes as input a secret key K, a nonce N, associated data A, and a message M to return a ciphertext C.
- The deterministic decryption algorithm $\mathtt{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \mathcal{M} \cup \{\perp\}$ takes as input a secret key K, a nonce N, associated data A, and a ciphertext C to return either a message in $\mathcal{M}$ or $\perp$ indicating that the ciphertext is invalid.

We denote the key space, the nonce space, the associated data space, and the message space of the scheme by $\mathcal{K}$, $\mathcal{N}$, $\mathcal{A}$, and $\mathcal{M}$, respectively. We assume that the algorithms are never queried on input not within these sets. A *symmetric encryption scheme* is analogously defined with the difference that its input does not admit associated data. In the security games, this entails that A is implicitly set to the empty string. In this work we focus on a specific class of encryption schemes, which we call *mirror-like*. These are encryption schemes where the encryption algorithm is an involution. Such schemes are fully determined by their encryption algorithm. Examples for mirror-like encryption schemes are the generic encryption scheme underlying the FGHF$'$ construction [14] as well as the sponge-based encryption schemes used in the AEAD schemes Slae [14] and Isap [15]. Besides these concrete schemes, instantiating block ciphers with encryption modes like CFB, OFB, and CTR also yield mirror-like encryption schemes.

A *message authentication code* $\text{MAC} = (\mathtt{T}, \mathtt{V})$ is a pair of efficient algorithms with an associated key space $\mathcal{K}$, domain $\mathcal{X}$, and tag length t such that:

- The deterministic tagging algorithm $\mathtt{T} : \mathcal{K} \times \mathcal{X} \to \{0,1\}^t$ takes as input a key K and a value X to return a tag T of size t.
- The deterministic verification algorithm $\mathtt{V} : \mathcal{K} \times \mathcal{X} \times \{0,1\}^t \to \{\top, \perp\}$ takes as input a key K, a value X, and a tag T to return either $\top$ indicating a valid input or $\perp$ otherwise.

It is required that for any $K, X \in \mathcal{K} \times \mathcal{X}$, if $T \leftarrow \mathtt{T}(K, X)$ then $\mathtt{V}(K, X, T) = \top$. Within this work we only consider *canonical* MACs which are implicitly defined by the tagging algorithm $\mathtt{T}$, i.e., the verification algorithm recomputes the tag of the message and accepts if the given tag equals the recomputed tag. We write $\text{MAC}[\mathcal{F}]$ to denote the canonical MAC built from a function $\mathcal{F}$.

2.3 Leakage Model

Our leakage model follows [5,14], building on *leakage resilience* as defined in [17]. The model follows the *only computation leaks information* assumption [23], i.e., only data that is processed during computation can leak information. For instance, encrypting a message with a certain key can not leak information about another key. Leakage is modelled by (deterministic and efficiently computable) functions from some predetermined set $\mathcal{L}$. Leakage of composite constructions is the composition of the underlying leakage functions. Thus, if primitive C is a composition of primitives A and B with leakage sets $\mathcal{L}_A$ and $\mathcal{L}_B$, then $\mathcal{L}_C = \mathcal{L}_A \times \mathcal{L}_B$ is the leakage set of C. In this work we focus on non-adaptive leakage, which we model by restricting $\mathcal{L}$ to be a singleton. Since the leakage depends entirely on the concrete device, the non-adaptive leakage model is suitable in practice, also argued by several other works [1,16,18,29,31].

We define the leakage resilience security notions that we need throughout this work. Following the blueprint by Barwell et al. [5], all notions are defined via security games where the adversary has access to one or more leakage oracle(s) which leak and one or more challenge oracle(s) which do not leak. According to [6], the former represent the power of the adversary while the latter model its goals in breaking the security of the scheme. Regarding the queries by the adversary, we follow [5] and say that an adversary *forwards* and *repeats* a query if it repeats a query across different oracles and the same oracle, respectively. For instance, querying the same tuple to the leakage encryption and challenge encryption is considered forwarding as is querying the output of an encryption oracle to a decryption oracle.

Non-Adaptive Leakage. All security notions below are defined following the style put forth in [5]. In particular, the permitted leakage functions are given by a set of leakage functions $\mathcal{L}$. While all our proofs hold in the general setting of adaptive leakage, we emphasise that we focus on non-adaptive leakage, i.e., any leakage set should be thought of as a singleton. This stems from the fact that an instantiation of the FGHF$'$ construction requires a leakage-resilient pseudorandom function which is unachievable in the adaptive leakage setting as discussed in [31], unless further restrictions are imposed on the leakage.

2.4 Security Notions

Leakage-Resilient Encryption. Regarding the restrictions of nonce selection by the adversary, we define *semi-nonce-respecting* adversaries. These are adversaries which are nonce-respecting, i.e., they never repeat a nonce, with respect to the challenge encryption oracle, but not with respect to the leakage encryption oracle. This follows the recent definition of misuse-resilience given in [4] and used for instance in [20]. Regarding the decryption oracles, note that there is no restriction imposed on how the nonces are selected.

For authenticated encryption schemes with associated data, we aim at leakage-resilient authenticated encryption (LAE) security. It is the counterpart of the security notion given by Rogaway [27], recast in the leakage setting by Barwell et al. [5].

Definition 1 (LAE **Security**) *Let* AEAD = $(\mathtt{E}, \mathtt{D})$ *be an authenticated encryption scheme with associated data and the game* LAE *be as defined in Fig. 1. For any nonce-respecting adversary $\mathcal{A}$ that never forwards or repeats queries to or from the oracles* Enc *and* Dec *and only makes encryption and decryption queries containing leakage functions in the respective sets $\mathcal{L}_{AE}$ and $\mathcal{L}_{VD}$, describing the leakage sets for authenticated encryption and verified decryption, its corresponding* LAE *advantage is given by:*

$$\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{AEAD}}(\mathcal{A}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) = 2\Pr\left[\mathsf{LAE}^{\mathcal{A}} \Rightarrow \mathrm{true}\right] - 1\,.$$

```
Game LAE
procedure Initialize
  $b \leftarrow_\$ \{0,1\};\ K \leftarrow_\$ \mathcal{K}$

procedure Enc$(N, A, M)$
  if $b = 0$
    return $C' \leftarrow_\$ \{0,1\}^{|\mathtt{E}(K,N,A,M)|}$
  return $C \leftarrow \mathtt{E}(K, N, A, M)$

procedure Finalize $(b')$
  return $(b' = b)$

procedure Dec$(N, A, C)$
  if $b = 0$
    return $\perp$
  return $M \leftarrow \mathtt{D}(K, N, A, C)$

procedure LEnc$(N, A, M, L)$
  $\Lambda \leftarrow L(K, N, A, M)$
  $C \leftarrow \mathtt{E}(K, N, A, M)$
  return $(C, \Lambda)$

procedure LDec$(N, A, C, L)$
  $\Lambda \leftarrow L(K, N, A, C)$
  $M \leftarrow \mathtt{D}(K, N, A, C)$
  return $(M, \Lambda)$
```

Fig. 1: Game used to define LAE security.

For symmetric encryption schemes we require IND-CPLA security as defined in [5], which corresponds to the classical notion of IND-CPA security enhanced with leakage.

Definition 2 (IND-CPLA **Security**) *Let* SE = $(\mathtt{E}, \mathtt{D})$ *be a symmetric encryption scheme and the game* INDCPLA *be as defined in Fig. 2. For any semi-nonce-respecting adversary $\mathcal{A}$ that never forwards or repeats queries to or from the oracle* Enc *and only makes encryption queries containing leakage functions in the set $\mathcal{L}_E$, its corresponding* IND-CPLA *advantage is given by:*

$$\mathbf{Adv}^{\mathsf{INDCPLA}}_{\mathrm{SE}}(\mathcal{A}, \mathcal{L}_E) = 2\Pr\left[\mathsf{INDCPLA}^{\mathcal{A}} \Rightarrow \mathrm{true}\right] - 1\,.$$

The N2 composition theorem in [5] requires a stronger variant called IND-aCPLA, where the 'a' stands for *augmented*. In this notion, the adversary also gets access to a leakage decryption oracle. The queries, however, are heavily restricted as it can only be queried on queries forwarded from the leakage encryption oracle LEnc.

```
Game INDCPLA
procedure Initialize
  $b \leftarrow_{\$} \{0,1\};\ K \leftarrow_{\$} \mathcal{K}$

procedure LEnc($N, M, L$)
  $\Lambda \leftarrow L(K, N, M)$
  $C \leftarrow \mathtt{E}(K, N, M)$
  return $(C, \Lambda)$

procedure Enc($N, M$)
  $C \leftarrow \mathtt{E}(K, N, M)$
  if $b = 0$
    return $C' \leftarrow_{\$} \{0,1\}^{|C|}$
  else
    return $C$

procedure Finalize ($b'$)
  return $(b' = b)$
```

Fig. 2: Game used to define IND-CPLA security.

Leakage-Resilient MACs and Function Families. For MACs, we target essentially the same security notion as in [5]. The difference is that we allow the adversary to forward queries between its leakage oracles, while [5] does not allow any forwarding from its leakage tagging oracle but forwarding from its leakage verification to its challenge oracle. Since the notions are very much akin, we write SUF-CMLA for our notion and SUF-CMLA* for the one from [5].

Definition 3 (SUF-CMLA **Security**) *Let* $\textsc{Mac} = (\mathtt{T}, \mathtt{V})$ *be a message authentication code and the game* SUFCMLA *be as defined in Fig. 3. For any adversary* $\mathcal{A}$ *that never forwards queries to or from the oracle* Vfy*, and only queries leakage functions to its oracles* LTag *and* LVfy *in the respective sets* $\mathcal{L}_T$ *and* $\mathcal{L}_V$*, its corresponding* SUF-CMLA *advantage is given by:*

$$\mathbf{Adv}_{\textsc{Mac}}^{\mathsf{SUFCMLA}}(\mathcal{A}, \mathcal{L}_T, \mathcal{L}_V) = 2\Pr\left[\mathsf{SUFCMLA}^{\mathcal{A}} \Rightarrow \text{true}\right] - 1\,.$$

```
Game SUFCMLA
procedure Initialize
  $b \leftarrow_{\$} \{0,1\};\ K \leftarrow_{\$} \mathcal{K}$

procedure Vfy($X, T$)
  if $b = 0$
    return $\perp$
  else
    $v \leftarrow \mathtt{V}(K, X, T)$
    return $v$

procedure LTag($X, L$)
  $\Lambda \leftarrow L(K, X)$
  $T \leftarrow \mathtt{T}(K, X)$
  return $(T, \Lambda)$

procedure LVfy($X, T, L$)
  $\Lambda \leftarrow L(K, X, T)$
  $v \leftarrow \mathtt{V}(K, X, T)$
  return $(v, \Lambda)$

procedure Finalize ($b'$)
  return $(b' = b)$
```

Fig. 3: Game used to define SUF-CMLA security.

For function families, we define both pseudorandomness and unpredictability under leakage. The former is well established in the literature, the latter was only recently introduced [14].

Definition 4 (LPRF **Security**) *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^t$ be a function family over the domain $\mathcal{X}$ and indexed by $\mathcal{K}$, and the game* LPRF *be as defined in Fig. 4. For any adversary $\mathcal{A}$ that never forwards or repeats queries to or from the oracle* F *and only queries leakage functions in the set $\mathcal{L}_F$, its corresponding* LPRF *advantage is given by:*

$$\mathbf{Adv}_{\mathcal{F}}^{\mathsf{LPRF}}(\mathcal{A}, \mathcal{L}_F) = 2\Pr\left[\mathsf{LPRF}^{\mathcal{A}} \Rightarrow \text{true}\right] - 1\,.$$

Removing the leakage oracle LF restores the classical notion of PRF security. We denote the corresponding game analogously to the other games by PRF (dropping the L for 'leakage'). We will use this game for our separation example in Section 3.

```
Game LPRF

procedure Initialize
  $b \leftarrow_\$ \{0,1\}$;  $K \leftarrow_\$ \mathcal{K}$

procedure LF$(X, L)$
  $y \leftarrow F(K, X)$
  $\Lambda \leftarrow L(K, X)$
  return $(y, \Lambda)$

procedure F$(X)$
  if $b = 0$
    return $y \leftarrow_\$ \{0,1\}^t$
  else
    return $F(K, X)$

procedure Finalize $(b')$
  return $(b' = b)$
```

Fig. 4: Game used to define LPRF security.

Definition 5 (LUF **Security**) *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^t$ be a function family over the domain $\mathcal{X}$ and indexed by $\mathcal{K}$, and the* LUF *game be as defined in Fig. 5. Then for any adversary $\mathcal{A}$ its corresponding* LUF *advantage is given by:*

$$\mathbf{Adv}_{\mathcal{F}}^{\mathsf{LUF}}(\mathcal{A}, \mathcal{L}_F) = \Pr\left[\mathsf{LUF}^{\mathcal{A}} \Rightarrow \text{true}\right].$$

A crucial difference between LUF and LPRF is that the former allows the adversary to obtain leakage for an input and still being able to win the game by predicting the output for this input while the latter does not allow such queries. This is exactly the difference that we exploit in our separation example.

```
Game LUF

procedure Initialize
  win $\leftarrow$ false;  $K \leftarrow_\$ \mathcal{K}$

procedure Lkg$(X, L)$
  $\Lambda \leftarrow L(K, X)$
  return $\Lambda$

procedure Finalize
  return (win)

procedure F$(X)$
  $\mathcal{S} \leftarrow_\cup X$
  $Y \leftarrow \mathcal{F}(K, X)$
  return $Y$

procedure Guess$(X, Y')$
  $Y \leftarrow \mathcal{F}(K, X)$
  if $X \notin \mathcal{S} \wedge Y = Y'$
    win $\leftarrow$ true
  return $(Y = Y')$
```

Fig. 5: Game used to define LUF security.

Pseudorandom Generator and Hash Functions. We make use of the following definition of a pseudorandom generator which enables the adversary to specify the output length (in bits) by querying it to the challenge oracle. The difference to [14] is that we stick to the single challenge case as opposed to their notion of multiple challenges.

Definition 6 (Pseudorandom Generators) *Let $\mathcal{G}\colon \mathcal{S} \times \mathbb{N} \to \{0,1\}^*$ be a pseudorandom generator with an associated seed space $\mathcal{S}$, and let the* PRG *game be as defined in Fig. 6. Then for any adversary $\mathcal{A}$, making exactly one query to* G*, its corresponding* PRG *advantage is given by:*

$$\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}) = 2\Pr\left[\mathsf{PRG}^{\mathcal{A}} \Rightarrow \text{true}\right] - 1\,.$$

```
Game PRG
procedure Initialize
    $b \leftarrow_\$ \{0,1\}$
procedure Finalize $(b')$
    return $(b' = b)$

procedure G$(L)$
    if $b = 0$
        $R \leftarrow_\$ \{0,1\}^L$
    else
        $S \leftarrow_\$ \mathcal{S}$
        $R \leftarrow \mathcal{G}(S, L)$
    return $R$
```

Fig. 6: Game used to define PRG security.

For a hash function $\mathcal{H}$ over a generic domain $\mathcal{X}$, we define its collision resistance below.

Definition 7 (Collision-Resistant Hash Functions) *Let $\mathcal{H}\colon \mathcal{X} \to \{0,1\}^w$ be a hash function. Then for any adversary $\mathcal{A}$ its corresponding advantage is given by:*

$$\mathbf{Adv}^{\mathrm{CR}}_{\mathcal{H}}(\mathcal{A}) = 2\Pr\left[\mathcal{H}(X_0) = \mathcal{H}(X_1) \wedge X_0 \neq X_1 \wedge X_0, X_1 \in \mathcal{X} \mid (X_0, X_1) \leftarrow \mathcal{A}\right]\,.$$

2.5 The FGHF′ Construction

Degabriele et al. [14] developed the FGHF′ construction, which allows to build a leakage-resilient AEAD scheme from four simple building blocks: two fixed-input-length functions $\mathcal{F}$ and $\mathcal{F}'$, a pseudorandom generator $\mathcal{G}$, and a hash function $\mathcal{H}$. The function $\mathcal{F}$ and the pseudorandom generator $\mathcal{G}$ build the encryption scheme $\mathrm{SE}[\mathcal{F}, \mathcal{G}]$ while the hash function $\mathcal{H}$ and the function $\mathcal{F}'$ build the MAC $\mathrm{MAC}[\mathcal{H}, \mathcal{F}']$. The construction is illustrated in Fig. 7 while the pseudocode is given in Fig. 8.

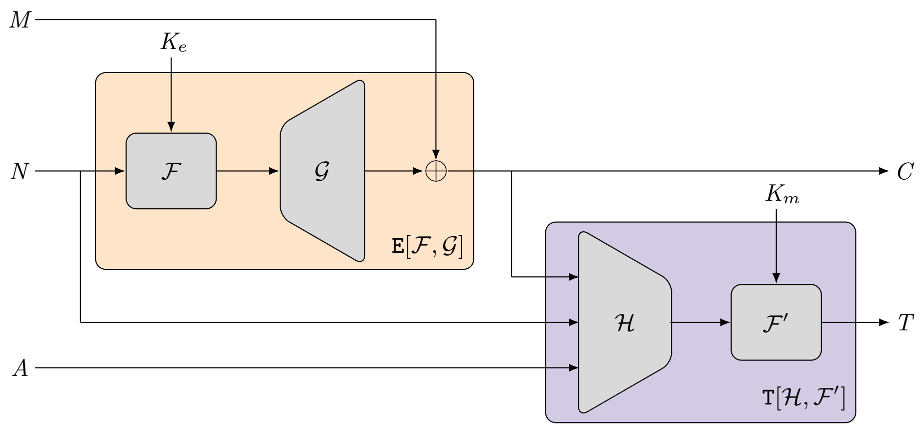

Fig. 7: Graphical representation of the FGHF′ construction. It corresponds to the N2 composition of $\mathrm{SE}[\mathcal{F}, \mathcal{G}]$ and $\mathrm{MAC}[\mathcal{H}, \mathcal{F}']$ which are in turn composed of a fixed-input-length LPRF $\mathcal{F}$, a PRG $\mathcal{G}$, a vector hash $\mathcal{H}$, and a fixed-input-length function $\mathcal{F}'$ that is both a LUF and an LPRF. The encryption and tagging algorithm of $\mathrm{SE}[\mathcal{F}, \mathcal{G}]$ and $\mathrm{MAC}[\mathcal{H}, \mathcal{F}']$ are $\mathtt{E}[\mathcal{F}, \mathcal{G}]$ and $\mathtt{T}[\mathcal{H}, \mathcal{F}']$, respectively.

```
$\mathtt{E}((K_e, K_m), N, A, M)$
    // Compute ciphertext using $\textsc{Se}[\mathcal{F}, \mathcal{G}]$
    $S \leftarrow \mathcal{F}(K_e, N)$
    $C_e \leftarrow \mathcal{G}(S, |M|) \oplus M$
    // Compute tag using $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$
    $H \leftarrow \mathcal{H}(N, A, C_e)$
    $T \leftarrow \mathcal{F}'(K_m, H)$
    return $C \leftarrow (C_e, T)$

$\mathtt{D}((K_e, K_m), N, A, (C_e, T))$
    $H \leftarrow \mathcal{H}(N, A, C_e)$
    $T' \leftarrow \mathcal{F}'(K_m, H)$
    if $T' = T$
        $S \leftarrow \mathcal{F}(K_e, N)$
        $M \leftarrow \mathcal{G}(S, |C_e|) \oplus C_e$
        return $M$
    return $\perp$
```

Fig. 8: Pseudocode of the FGHF$'$ construction.

The notable feature of the construction is that only the fixed-input-length functions have to be leakage-resilient, while the pseudorandom generator and the hash function can be instantiated with off-the-shelf primitives from the literature. The security implications, which illustrate the main result from [14], are displayed in Fig. 9. Note the special structure of the FGHF$'$ construction, that is, $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ being a mirror-like encryption scheme and $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ being a canonical MAC (considering the composition of $\mathcal{H}$ and $\mathcal{F}'$ a function with variable-input-length). Combined with our leakage model, we conclude that the leakage sets $\mathcal{L}_E$ and $\mathcal{L}_D$ for $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ are equal as are the leakage sets $\mathcal{L}_T$ and $\mathcal{L}_V$ for $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$, i.e., $\mathcal{L}_E = \mathcal{L}_D = \mathcal{L}_F \times \mathcal{L}_G$ and $\mathcal{L}_T = \mathcal{L}_V = \mathcal{L}_H \times \mathcal{L}_{F'}$. Here, $\mathcal{L}_F$, $\mathcal{L}_G$, $\mathcal{L}_H$, and $\mathcal{L}_{F'}$ are the leakage sets of the underlying components $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{F}'$. The very same is implicitly assumed in [14]. Likewise, we obtain the leakage sets $\mathcal{L}_{AE} = \mathcal{L}_E \times \mathcal{L}_T = \mathcal{L}_F \times \mathcal{L}_G \times \mathcal{L}_H \times \mathcal{L}_{F'} = \mathcal{L}_D \times \mathcal{L}_V = \mathcal{L}_{VD}$ for the resulting AEAD scheme.

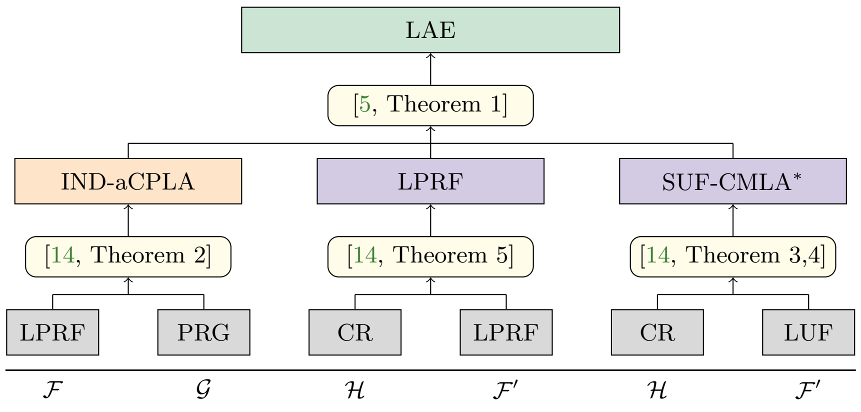

Fig. 9: Security implications for the FGHF$'$ construction from [14, Theorem 6]. Note that we do not give the formal definition of IND-aCPLA and SUF-CMLA* as we use slightly different notions.

3 Unpredictability and Pseudorandomness under Leakage

Along with the FGHF$'$ construction, Degabriele et al. [14] introduce a security notion for unpredictability of functions under leakage. They prove the existence of functions that achieve both unpredictability and pseudorandomness under leakage. Regarding the relation between these notions, they claim them to be incomparable, without giving a clear justification or a proof for this statement. We confirm this by providing a separation example which proves that the notions do not imply one another. Therefore, we give two functions: under leakage, the first function is unpredictable but not pseudorandom, while the second function is pseudorandom but not unpredictable. For both functions, we assume a function which, under leakage, is both unpredictable and pseudorandom. Note that this assumption is valid as the existence of such functions has been shown in [14] for the sponge-based instantiation $\textsc{Slae}$ of the FGHF$'$ construction.

3.1 Under Leakage: Unpredictability $\not\Rightarrow$ Pseudorandomness

We start with the simple case, that is, a function which is unpredictable but not pseudorandom under leakage.

Construction 8 *Let $\mathcal{F}_*: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^t$ be a function. Define the function*

$$\begin{aligned}\mathcal{F}: \{0,1\}^k \times \{0,1\}^n &\to \{0,1\}^{t+1}\\ \mathcal{F}(K,X) &\mapsto 0 \parallel \mathcal{F}_*(K,X)\,.\end{aligned}$$

Lemma 9 *Let $\mathcal{F}_*$ be a function that is both a* LUF *and an* LPRF *and $\mathcal{F}$ be the function constructed from $\mathcal{F}_*$ according to Construction 8. It holds that $\mathcal{F}$ is a* LUF *but not an* LPRF*.*

Proof. The function $\mathcal{F}$ is LUF as any LUF adversary against $\mathcal{F}$ can easily be transformed into a LUF adversary against $\mathcal{F}_*$. On the other hand, the leading 0 of any output makes it easy to distinguish the function from a random function. If, after several queries, an output starting with 1 is observed, the adversary outputs 0 (indicating ideal), otherwise, it outputs 1 (indicating real). □

3.2 Under Leakage: Pseudorandomness $\not\Rightarrow$ Unpredictability

Now we address the complex part of the separation example and show that there are functions which are pseudorandom but not unpredictable under leakage.

Construction 10 *Let $\mathcal{F}_O: \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^t$ and $\mathcal{F}_I: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^s$ be functions. The subscripts O and I indicate the outer and inner function, respectively. Define the function*

$$\begin{aligned}\mathcal{F}: \{0,1\}^k \times \{0,1\}^n &\to \{0,1\}^t\\ \mathcal{F}(K,X) &\mapsto \mathcal{F}_O(\mathcal{F}_I(K,X),X)\,.\end{aligned}$$

The idea of Construction 10 is as follows. It uses some *master key* K and, for each input X, it derives a *session key* using the inner function $\mathcal{F}_I$, i.e., $K_s = \mathcal{F}_I(K,X)$. The output Y is generated by the outer function $\mathcal{F}_O$ using the session key K_s and input X. The lemma below shows that the construction is pseudorandom but not unpredictable under leakage.

Lemma 11 *Let $\mathcal{F}_O$ and $\mathcal{F}_I$ be two functions and $\mathcal{F}$ be the function constructed from $\mathcal{F}_O$ and $\mathcal{F}_I$ according to Construction 10. Suppose $\mathcal{F}_I$ is both a* LUF *and an* LPRF *and $\mathcal{F}_O$ is* PRF*. Then $\mathcal{F}$ is an* LPRF *but not a* LUF*.*

Proof. We first show that $\mathcal{F}$ is an LPRF. For simplicity, we restrict the adversary to a single challenge query and argue at the end how it can be lifted to multiple challenge queries. We make use of the games G_0, G_1, and G_2 displayed in Fig. 10. The games are constructed such that G_0 is equal to LPRF with secret bit $b = 1$ and G_2 is equal to LPRF with secret bit $b = 0$. Recall that the leakage set $\mathcal{L}_F$ of $\mathcal{F}$ is the Cartesian product of the leakage sets $\mathcal{L}_O$ of $\mathcal{F}_O$ and $\mathcal{L}_I$ of $\mathcal{F}_I$. Reformulation to the adversarial advantage yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A},\mathcal{L}_F) &= 2\Pr\left[\mathsf{LPRF}^{\mathcal{A}} \Rightarrow \text{true}\right] - 1\\ &= 2\left(\Pr\left[\mathsf{LPRF}^{\mathcal{A}} \Rightarrow \text{true} \wedge b = 1\right] + \Pr\left[\mathsf{LPRF}^{\mathcal{A}} \Rightarrow \text{true} \wedge b = 0\right]\right) - 1\\ &= 2\left(\Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 1 \wedge b = 1\right] + \Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 0 \wedge b = 0\right]\right) - 1\\ &= 2\left(\Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 1\right]\Pr[b=1] + \Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 0 \,|\, b = 0\right]\Pr[b=0]\right) - 1\\ &= \Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 1\right] + \Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 0 \,|\, b = 0\right] - 1\\ &= \Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{LPRF}} \Rightarrow 1 \,|\, b = 0\right]\\ &= \Pr\left[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1\right]\,. \end{aligned} \tag{1}$$

For the game hop between G_0 and G_1, we construct an LPRF adversary $\mathcal{A}_{lprf}$ against $\mathcal{F}_I$ as follows. When $\mathcal{A}$ makes its query X to F, $\mathcal{A}_{lprf}$ queries X to its own challenge oracle to obtain the session key K_s, computes $Y \leftarrow \mathcal{F}_O(K_s,X)$, and sends Y back to $\mathcal{A}$. For leakage queries $(X,(L_O,L_I))$ by $\mathcal{A}$, $\mathcal{A}_{lprf}$ queries its own leakage oracle on (X,L_I) to obtain (K_s,Λ_I), computes $Y \leftarrow \mathcal{F}_O(K_s,X)$ and $\Lambda_O \leftarrow L_O(K_s,X)$,

```
Initialize
  $K \leftarrow_\$ \{0,1\}^k$

procedure $\mathsf{LF}(X, (L_O, L_I))$
  $K_s \leftarrow \mathcal{F}_I(K, X)$
  $\Lambda_I \leftarrow L_I(K, X)$
  $Y \leftarrow \mathcal{F}_O(K_s, X)$
  $\Lambda_O \leftarrow L_O(K_s, X)$
  return $(Y, (\Lambda_O, \Lambda_I))$

procedure $\mathsf{F}(X)$ in $\mathsf{G}_0$
  $K_s \leftarrow \mathcal{F}_I(K, X)$
  return $Y \leftarrow \mathcal{F}_O(K_s, X)$

procedure $\mathsf{F}(X)$ in $\mathsf{G}_1$
  $K_s \leftarrow_\$ \{0,1\}^s$
  return $Y \leftarrow \mathcal{F}_O(K_s, X)$

procedure $\mathsf{F}(X)$ in $\mathsf{G}_2$
  return $Y \leftarrow_\$ \{0,1\}^t$
```

Fig. 10: Games G_0, G_1, and G_2 used in the proof of Lemma 11. The games share the leakage oracle LF, while each game uses its own challenge oracle as described.

and sends $(Y, (\Lambda_O, \Lambda_I))$ back to $\mathcal{A}$. It is easy to see that $\mathcal{A}_{lprf}$ perfectly simulates G_0 or G_1 for $\mathcal{A}$ depending on its own challenge. Also all queries by $\mathcal{A}_{lprf}$ are permitted as it queries exactly the same values as $\mathcal{A}$. Hence we conclude with

$$\Pr\left[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1\right] \leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}_I}(\mathcal{A}_{lprf}, \mathcal{L}_I)\,. \tag{2}$$

For the game hop between G_1 and G_2, we construct the following PRF adversary $\mathcal{A}_{prf}$ against $\mathcal{F}_O$. At the start, $\mathcal{A}_{prf}$ samples a random master key K which it will use for the leakage queries by $\mathcal{A}$. Whenever $\mathcal{A}$ queries $(X, (L_O, L_I))$ to its leakage oracle, $\mathcal{A}_{prf}$ (locally) computes $K_s \leftarrow \mathcal{F}_I(K, X)$, $\Lambda_I \leftarrow L_I(K, X)$, $Y \leftarrow \mathcal{F}_O(K_s, X)$, and $\Lambda_O \leftarrow L_O(K_s, X)$, and sends $(Y, (\Lambda_O, \Lambda_I))$ to $\mathcal{A}$. For the challenge query X by $\mathcal{A}$, $\mathcal{A}_{prf}$ forwards the query to its own challenge oracle and the response back to $\mathcal{A}$. It is again easy to see that $\mathcal{A}_{prf}$ perfectly simulates the games G_1 and G_2 for $\mathcal{A}$ depending on its own challenge. The significant feature is that $\mathcal{A}_{prf}$ can simulate the leakage oracle for $\mathcal{A}$ locally, which is why we only need PRF security as opposed to LPRF security. We conclude with

$$\Pr\left[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1\right] \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{F}_O}(\mathcal{A}_{prf})\,. \tag{3}$$

Inserting (2) and (3) in (1) yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}, \mathcal{L}_F) &= \Pr\left[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 1\right] \\ &\leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}_I}(\mathcal{A}_{lprf}, \mathcal{L}_I) + \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{F}_O}(\mathcal{A}_{prf})\,.\end{aligned}$$

We briefly discuss how the proof can be adapted if $\mathcal{A}$ is allowed to make multiple challenge queries. The first part works exactly the same, that is, $\mathcal{A}_{lprf}$ forwards the query X to get the session key K_s and computes $Y \leftarrow \mathcal{F}_O(K_s, X)$. For the second part, there is a subtle issue why the same reduction does not work if $\mathcal{A}$ makes multiple challenge queries. In G_1, $\mathcal{A}$ expects that every query uses a fresh session key sampled uniformly at random, while the key used in the game PRF is fixed, thus it can not simulate the correct game for $\mathcal{A}$. Instead, the game hop can be lifted to multiple challenge queries via a straightforward hybrid argument, where $\mathcal{A}_{prf}$ answers the first $i-1$ queries with random values, the i-th query using its own challenge oracle just as described for the single challenge case, and the last $q-i$ queries with $\mathcal{F}_O(K_s, X)$ for a randomly chosen session key K_s. This induces a factor q, the number of challenge queries, into the bound above.

It remains to show that $\mathcal{F}$ is not a LUF. We construct the following LUF adversary. It queries its leakage oracle Lkg on a randomly chosen input X, leaking the session key K_s. Given the session key, it computes $Y \leftarrow \mathcal{F}_O(K_s, X)$ and sends (X, Y) to its challenge oracle Guess, which will set the winning flag to true. □

Our LUF adversary exploits the fact that the game LUF allows to obtain leakage for the input for which the output is predicted. Hence, the adversary leaks the session key K_s for some input X, which enables it to perfectly predict the output. Note that this does not enable the adversary to predict an output for a different input. Our LUF adversary bypasses the LUF security of the inner function $\mathcal{F}_I$ and attacks the outer function $\mathcal{F}_O$ instead. Regarding the LPRF security, observe the following. The LPRF adversary is also able to obtain a session key K_s through its leakage oracle. However, this session key is only valid for the queried input and the game LPRF does not allow to query this input to the challenge oracle, which would make the notion unachievable anyway. We essentially show that it is sufficient to secure the master key K by deriving the session keys using an inner function with strong security guarantees (LPRF and LUF in our case).

We believe this result to be of independent interest as it shows that extra caution is judicious in the leakage setting. Our constructions show how well-known and established results from the leak-free setting, like the equivalence between pseudorandomness and unpredictability, do not necessarily remain valid in the leakage setting.

4 Leakage Resilience of the N2 Construction

In the previous section we established that the security notions LUF and LPRF are incomparable. This entails that any instantiation of the FGHF′ construction has to prove LPRF and LUF security separately. Considering the instantiation SLAE [14], proving these notions is the most complex part of the work. Thus, we now turn our attention towards removing the requirement of LUF security. As argued in [14], LUF security is required to build a secure MAC according to the composition theorem for the N2 construction [5] (see also Fig. 9). Recall that Degabriele et al. [14] prove that the encryption scheme $\mathrm{SE}[\mathcal{F},\mathcal{G}]$ and the MAC $\mathrm{MAC}[\mathcal{H},\mathcal{F}']$, underlying the FGHF′ construction, achieve the security notion required by the N2 composition theorem by Barwell et al. [5]. However, Barwell et al. prove their composition for arbitrary encryption schemes and MACs, while Degabriele et al. focus on a mirror-like encryption scheme and a canonical MAC. In this section, we recast the composition theorem for the N2 construction to the case of mirror-like encryption schemes and canonical MACs.

The theorem below shows that the N2 composition of a mirror-like encryption scheme and a canonical MAC is LAE-secure if the underlying components are. For ease of exposition, we give the full proof for our setting. Subsequently, we discuss how our proof differs from the one given in [5].

Theorem 12 (LAE Security of the N2 Construction) *Let $\mathrm{SE} = (\mathtt{E},\mathtt{D})$ be a mirror-like symmetric encryption scheme with associated leakage sets $(\mathcal{L}_E,\mathcal{L}_D)$ and $\mathrm{MAC} = (\mathtt{T},\mathtt{V})$ be a canonical MAC with associated leakage sets $(\mathcal{L}_T,\mathcal{L}_V)$. Let N2 be the composition of SE and MAC as displayed in Fig. 7 with associated leakage sets $\mathcal{L}_{AE} = \mathcal{L}_E \times \mathcal{L}_T$ and $\mathcal{L}_{VD} = \mathcal{L}_D \times \mathcal{L}_V$. For any nonce-respecting adversary $\mathcal{A}_{ae}$ against N2 there exists a semi-nonce-respecting IND-CPLA adversary $\mathcal{A}_{se}$ against SE, an LPRF adversary $\mathcal{A}_{lprf}$ against $\mathtt{T}$, and a SUF-CMLA adversary $\mathcal{A}_{mac}$ against MAC such that:*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{N2}}(\mathcal{A}_{ae},\mathcal{L}_{AE},\mathcal{L}_{VD}) \le \mathbf{Adv}^{\mathsf{INDCPLA}}_{\mathrm{SE}}(\mathcal{A}_{se},\mathcal{L}_E) + \mathbf{Adv}^{\mathsf{LPRF}}_{\mathtt{T}}(\mathcal{A}_{lprf},\mathcal{L}_T)\\ + \mathbf{Adv}^{\mathsf{SUFCMLA}}_{\mathrm{MAC}}(\mathcal{A}_{mac},\mathcal{L}_T,\mathcal{L}_V)\,.\end{aligned}$$

Proof. Since SE is a mirror-like encryption scheme and MAC is a canonical MAC, it holds that $\mathcal{L}_E = \mathcal{L}_D$ and $\mathcal{L}_T = \mathcal{L}_V$. This immediately implies that the leakage sets $\mathcal{L}_{AE}$ and $\mathcal{L}_{VD}$ of N2 are identical. We us a sequence of games $\mathsf{G}_0,\dots,\mathsf{G}_3$ shown in Fig. 11. Game G_0 is the game LAE instantiated with N2 with secret bit fixed to 1. In game G_1, the decryption oracle is changed to reject any queried ciphertext. Game G_2 and G_3 are like G_1, except for the following differences. Game G_2 generates challenge ciphertext by sampling it at random while but still computing the real tag of this ciphertext. In game G_3 both the ciphertext and the tag are sampled at random. Hence G_3 equals the game LAE instantiated with N2 with secret bit fixed to 0. Using a simple reformulation to the adversarial advantage, similar to (1), yields

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{N2}}(\mathcal{A}_{ae},\mathcal{L}_{AE},\mathcal{L}_{VD}) &= \Pr\left[\mathcal{A}_{ae}^{\mathsf{LAE}} \Rightarrow 1 \,|\, b=1\right] - \Pr\left[\mathcal{A}_{ae}^{\mathsf{LAE}} \Rightarrow 1 \,|\, b=0\right]\\ &= \Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_0} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_3} \Rightarrow 1\right]\\ &= \sum_{i=1}^{3}\left(\Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_{i-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_i} \Rightarrow 1\right]\right)\,. \qquad (4)\end{aligned}$$

Below we bound each of the game hops. The hop from G_0 to G_1 is bound by the SUF-CMLA security of the underlying MAC MAC, the hop from G_1 to G_2 by the IND-CPLA security of the underlying encryption scheme SE, and the hop from G_2 to G_3 by the LPRF security of the tagging algorithm $\mathtt{T}$.

Let us start with the adversarial advantage between games G_0 and G_1. We construct the following SUF-CMLA adversary $\mathcal{A}_{mac}$. It samples a random key K_e for the encryption scheme SE. Queries

```
procedure Initialize
  $K_e, K_m \leftarrow_\$ \mathcal{K}$

procedure Enc$(N, A, M)$
  $C_e \leftarrow \mathtt{E}(K_e, N, M)$
  $T \leftarrow \mathtt{T}(K_m, N, A, C_e)$
  return $C \leftarrow (C_e, T)$

procedure LEnc$(N, A, M, L)$
  parse $L$ as $(L_e, L_t)$
  $C_e \leftarrow \mathtt{E}(K_e, N, M)$
  $T \leftarrow \mathtt{T}(K_m, N, A, C_e)$
  $C \leftarrow (C_e, T)$
  $\Lambda_e \leftarrow L_e(K_e, N, M)$
  $\Lambda_t \leftarrow L_t(K_m, N, A, C_e)$
  $\Lambda \leftarrow (\Lambda_e, \Lambda_t)$
  return $(C, \Lambda)$

procedure Dec$(N, A, C)$ in $\mathsf{G}_0$ and $\boxed{\mathsf{G}_1}$
  parse $C$ as $(C_e, T)$
  $v \leftarrow \mathtt{V}(K_m, N, A, C_e, T)$
  if $v = \perp$
    return $\perp$
  $\boxed{\textbf{return } \perp}$
  return $M \leftarrow \mathtt{D}(K_e, N, C_e)$

procedure LDec$(N, A, C, L)$
  parse $L$ as $(L_d, L_v)$
  parse $C$ as $(C_e, T)$
  $v \leftarrow \mathtt{V}(K_m, N, A, C_e, T)$
  $\Lambda_v \leftarrow L_v(K_m, N, A, C_e, T)$
  if $v = \perp$
    return $(\perp, \Lambda_v)$
  $M \leftarrow \mathtt{D}(K_e, N, C_e)$
  $\Lambda_d \leftarrow L_d(K_e, N, C_e)$
  return $(M, \Lambda_d, \Lambda_v)$
```

```
procedure Enc$(N, A, M)$ in $\mathsf{G}_2$
  $C_e \leftarrow_\$ \{0,1\}^{|\mathtt{E}(K_e,N,M)|}$
  $T \leftarrow \mathtt{T}(K_m, N, A, C_e)$
  return $C \leftarrow (C_e, T)$

procedure Enc$(N, A, M)$ in $\mathsf{G}_3$
  $C_e \leftarrow_\$ \{0,1\}^{|\mathtt{E}(K_e,N,M)|}$
  $T \leftarrow_\$ \{0,1\}^{|\mathtt{T}(K_m,N,A,C_e)|}$
  return $C \leftarrow (C_e, T)$
```

Fig. 11: Games $\mathsf{G}_0, \ldots, \mathsf{G}_3$ used to prove Theorem 12. The oracles LEnc and LDec are identical across the games. Game G_1 is the same G_0, except that it includes the boxed code. Games G_2 and G_3 are like G_1, except for the oracle Enc which are displayed at the bottom.

(N, A, M) to Enc are answered by $\mathcal{A}_{mac}$ as follows. It computes the ciphertext $C_e \leftarrow \mathtt{E}(K_e, N, M)$, obtains the tag T by invoking its oracle LTag on $((N, A, C_e), \emptyset)$[2], and sends the ciphertext $C \leftarrow (C_e, T)$ back to $\mathcal{A}_{ae}$. Leakage encryption queries $(N, A, M, (L_e, L_t))$ are processed as follows. The ciphertext $C_e \leftarrow \mathtt{E}(K_e, N, M)$ and corresponding leakage $\Lambda_e \leftarrow L_e(K_e, N, M)$ are computed locally by $\mathcal{A}_{mac}$. Subsequently, it queries its leakage oracle LTag on $((N, A, C_e), L_t)$ to obtain (T, Λ_t) and sends back $C \leftarrow (C_e, T)$ and $\Lambda \leftarrow (\Lambda_e, \Lambda_t)$ to $\mathcal{A}_{ae}$. For any leakage decryption query $(N, A, (C_e, T), (L_d, L_v))$, $\mathcal{A}_{mac}$ forwards $((N, A, C_e), T, L_v)$ to its leakage oracle LVfy to obtain (V, Λ_v), which it forwards to $\mathcal{A}_{ae}$ if $V = \bot$. Otherwise, i.e., $V = \top$, $\mathcal{A}_{mac}$ computes $M \leftarrow \mathtt{D}(K_e, N, C_e)$ and $\Lambda_d \leftarrow L_e(K_e, N, C_e)$, and sends back $(M, \Lambda_d, \Lambda_v)$ to $\mathcal{A}_{ae}$. Whenever $\mathcal{A}_{ae}$ queries its oracle Dec on $(N, A, (C_e, T))$, $\mathcal{A}_{mac}$ forwards $((N, A, C_e), T)$ to its oracle Vfy. If the response of Vfy is $\bot$, it forwards it to $\mathcal{A}_{ae}$, otherwise, it computes $M \leftarrow \mathtt{D}(K_e, N, C_e)$ and sends it to $\mathcal{A}_{ae}$.

Clearly, $\mathcal{A}_{mac}$ queries its leakage oracles LTag and LVfy only on the permissive functions, as $\mathcal{A}_{ae}$ does. $\mathcal{A}_{mac}$ does also not make any prohibited query, as it invokes its challenge oracle Vfy if and only if $\mathcal{A}_{ae}$ makes a query to its challenge decryption oracle Dec which never forwards any query to or from it.

Recall that the difference between G_0 and G_1 is that the former implements the real decryption oracle while the latter rejects any decryption query. Conditioned on the secret bit b of $\mathsf{SUFCMLA}$ being 0, $\mathcal{A}_{mac}$ never decrypts C_e, hence it perfectly simulates G_1 for $\mathcal{A}_{ae}$. Likewise, if $b = 1$, $\mathcal{A}_{mac}$ only decrypts if the tag T is valid, thus it perfectly simulates G_0 for $\mathcal{A}_{ae}$. Hence we conclude with

$$\Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_0} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_1} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathrm{MAC}}^{\mathsf{SUFCMLA}}(\mathcal{A}_{mac}, \mathcal{L}_T, \mathcal{L}_V)\,. \tag{5}$$

For the remaining game hops, note that the oracle Dec rejects any ciphertext irrespective of the validity of the tag which is why we omit it in the description as every reduction simply responds with $\bot$.

For the game hop between G_1 and G_2, we construct an IND-CPLA adversary $\mathcal{A}_{se}$ as follows. It generates a key K_m for MAC and runs the adversary $\mathcal{A}_{ae}$ answering its queries as follows. For leakage queries $(N, A, M, (L_e, L_t))$ to LEnc it passes (N, M, L_e) to its own oracle LEnc and obtains (C_e, Λ_e) in return. It computes the tag $T \leftarrow \mathtt{T}(K_m, N, A, C_e)$, corresponding leakage $\Lambda_t \leftarrow L_t(K_m, N, A, C_e)$, and sends $((C_e, T), (\Lambda_e, \Lambda_t))$ back to $\mathcal{A}_{ae}$. For leakage queries $(N, A, (C_e, T), (L_d, L_v))$ to LDec, $\mathcal{A}_{se}$ first computes $V \leftarrow \mathtt{V}(K_m, (N, A, C_e), T)$ and $\Lambda_v \leftarrow L_v(K_m, N, A, C_e, T)$. If $V = \bot$, it sends $(\bot, \Lambda_v)$ back to $\mathcal{A}_{ae}$. If $V = \top$, it forwards (N, M, L_e) to its own leakage encryption oracle LEnc to obtain (M, Λ_d)[3] and sends $(M, (\Lambda_d, \Lambda_v))$ back to $\mathcal{A}_{ae}$. Queries (N, A, M) to Enc are handled by obtaining C_e from its own challenge encryption oracle invoked with (N, M), computing the tag $T \leftarrow \mathtt{T}(K_m, N, A, C_e)$, and sending (C_e, T) back to $\mathcal{A}_{ae}$.

Since $\mathcal{A}_{ae}$ queries its leakage oracles only on functions in the corresponding leakage set, so does $\mathcal{A}_{se}$. Every challenge encryption query by $\mathcal{A}_{ae}$ entails that $\mathcal{A}_{se}$ invokes its challenge encryption query. Likewise, every leakage query, either encryption or decryption, leads to a leakage encryption query by $\mathcal{A}_{se}$. As a valid LAE adversary, $\mathcal{A}_{ae}$ does not forward queries from challenge to leakage oracles or vice versa, as does $\mathcal{A}_{se}$. Note further that $\mathcal{A}_{se}$ is semi-nonce-respecting. This follows from $\mathcal{A}_{se}$ simulating both leakage oracles of $\mathcal{A}_{ae}$ using its leakage encryption oracle and $\mathcal{A}_{ae}$ being nonce-respecting.

It is easy to see that $\mathcal{A}_{se}$ perfectly simulates either G_1 or G_2 for $\mathcal{A}_{se}$. The games differ in the ciphertext part C_e generated by Enc. In G_1 it is the encryption of the message M, while it is a random bit string in G_2. By setting C_e to the output of its own challenge oracle, $\mathcal{A}_{se}$ simulates G_1 and G_2 for $\mathcal{A}_{ae}$ conditioned on the secret bit b of the game $\mathsf{INDCPLA}$ being 1 and 0, respectively. It holds that

$$\Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_1} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_2} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathrm{SE}}^{\mathsf{INDCPLA}}(\mathcal{A}_{se}, \mathcal{L}_E)\,. \tag{6}$$

Finally, we construct the following LPRF adversary $\mathcal{A}_{lprf}$ to bound the adversarial advantage between G_2 and G_3. It generates a key K_e for the underlying encryption scheme. Leakage encryption queries $(N, A, M, (L_e, L_t))$ are processed by locally computing $C_e \leftarrow \mathtt{E}(K_e, N, M)$ and $\Lambda_e \leftarrow L_e(K_e, N, M)$, invoking LF on $((N, A, C_e), L_t)$ to obtain (T, Λ_t), and sending $((C_e, T), (\Lambda_e, \Lambda_t))$ back to $\mathcal{A}_{ae}$. For leakage decryption queries $(N, A, (C_e, T), (L_d, L_v))$, $\mathcal{A}_{lprf}$ sends $((N, A, C_e), L_v)$ to its leakage oracle LF to obtain (T', Λ_v). If $T \neq T'$, $\mathcal{A}_{lprf}$ sends $(\bot, \Lambda_v)$ to $\mathcal{A}_{ae}$. Otherwise, $\mathcal{A}_{lprf}$ computes locally $M \leftarrow \mathtt{D}(K_e, N, C_e)$ and $\Lambda_d \leftarrow L_d(K_e, N, C_e)$, and sends $(M, (\Lambda_d, \Lambda_v))$ to $\mathcal{A}_{ae}$. For queries (N, A, M) that $\mathcal{A}_{ae}$ makes to its challenge encryption oracle Enc, $\mathcal{A}_{lprf}$ samples a random bit string C_e of appropriate length, invokes its challenge oracle F on (N, A, C_e) to obtain T, and sends (C_e, T) back to $\mathcal{A}_{ae}$.

Recall that the difference between G_2 and G_3 is how the tag T is generated. In G_2 it is the real tag computed on a random ciphertext, in G_3 it is a random bit string. By construction, $\mathcal{A}_{lprf}$ perfectly simulates G_2 and G_3 if its own challenge bit b (from the game LPRF) is equal to 1 and 0, respectively.

[2] $\mathcal{A}_{mac}$ does not submit a leakage function, as it simulates a challenge oracle for $\mathcal{A}_{ae}$.
[3] Note that $\mathtt{E}(K, N, C) = \mathtt{D}(K, N, C)$.

Every challenge (leakage) query by $\mathcal{A}_{lprf}$ stems from a challenge (leakage) query by $\mathcal{A}_{ae}$. As $\mathcal{A}_{ae}$ does not forward queries between its challenge and leakage oracles neither does $\mathcal{A}_{lprf}$. Hence we conclude that $\mathcal{A}_{lprf}$ is a valid LPRF adversary against T.

$$\Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_2} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{ae}^{\mathsf{G}_3} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{T}}^{\mathsf{LPRF}}(\mathcal{A}_{lprf}, \mathcal{L}_T)\,. \tag{7}$$

Inserting (5), (6), and (7) in (4) gives the desired result. □

We will now go into the differences between our proof and the proof from [5]. In [5], the first game hop differs in that it also changes the leakage decryption oracle LDec. The change is such that any leakage decryption query which are not forwarded from the leakage encryption oracle is rejected by returning $\perp$. In [5], this change is necessary in order to bound the second game hop with the security of the underlying encryption scheme. To detect the difference, the LAE adversary $\mathcal{A}_{ae}$ has to submit a (fresh) valid ciphertext to LDec as an invalid ciphertext would be rejected anyway. This entails that $\mathcal{A}_{ae}$ has generated a (fresh) valid tag for this ciphertext, which the reduction will use to distinguish whether its challenge oracle implements the verification algorithm or $\perp$. Since the leakage decryption oracle is simulated via the leakage verification oracle, the reduction has to forward this leakage query to its own challenge oracle to distinguish between the real and the ideal world. This is exactly the query which prevents building such a MAC from a function which is pseudorandom and ultimately led to the introduction of LUF security by Degabriele et al. [14].

The next two game hops are the same as in our proof, except that the leakage decryption oracle does not decrypt any fresh ciphertext due to the change in the first game hop. This restriction allows to bound the second game hop by the IND-aCPLA security of the underlying encryption scheme, as the only queries that can not be answered with the oracle from the game $\mathsf{INDaCPLA}$ (decryption of fresh ciphertext) are answered with $\perp$. In our case of mirror-like encryption schemes, this issue does not arise if the scheme is secure with respect to semi-nonce-respecting adversaries in which case we only need IND-CPLA security as forwarded leakage decryption queries are answered like fresh queries.

The third game hop is essentially the same, again only differing in the leakage decryption oracle. Since the LPRF adversary simulates the encryption-related part of the game locally, this difference is trivial.

Finally, Barwell et al. [5] have a fourth game hop. In this game hop, where the challenge oracles are already idealised, they merely revert the change of the leakage decryption oracle from the first game hop in order to end up with the idealised game, that is LAE with secret bit 0. Since we never change any leakage oracle throughout our proof, we do not need this additional game hop.

5 Leakage Resilience of the FGHF′ Construction

Having established the leakage resilience of the N2 composition for mirror-like encryption schemes and canonical MACs, we turn our attention towards the FGHF′ construction. Since our recast composition theorem imposes different security notions for the encryption scheme and the MAC, it remains to show that the encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ and the MAC $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$ of the FGHF′ construction achieve these notions. In Section 5.1 we show that we can build a SUF-CMLA-secure MAC from a function which is pseudorandom under leakage. Combined with the result of Degabriele et al. [14] we obtain the SUF-CMLA security of $\textsc{Mac}[\mathcal{H}, \mathcal{F}']$. In Section 5.2, we show that the encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ (proven IND-aCPLA-secure against nonce-respecting adversaries by Degabriele et al. [14]) achieves IND-CPLA security against semi-nonce-respecting adversaries.

5.1 Building Leakage-Resilient MACs from LPRFs

The following theorem shows that we can construct a SUF-CMLA-secure MAC from a function that is an LPRF. The difference to [14] is that our security notion does not allow the adversary to forward queries from its leakage oracle to its challenge oracle. The proof is given in Appendix A.1.

Theorem 13 *Let $\mathcal{F}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^t$ be a function family with associated leakage set $\mathcal{L}_F$, and let $\textsc{Mac}[\mathcal{F}]$ be the corresponding canonical MAC with associated leakage sets $\mathcal{L}_T$, $\mathcal{L}_V$ where $\mathcal{L}_F = \mathcal{L}_T = \mathcal{L}_V$. Then, for any* SUF-CMLA *adversary $\mathcal{A}_{mac}$ against* $\textsc{Mac}[\mathcal{F}]$ *which makes q queries to* Vfy, *there exists an adversary $\mathcal{A}_{lprf}$ against $\mathcal{F}$ such that:*

$$\mathbf{Adv}_{\textsc{Mac}[\mathcal{F}]}^{\mathsf{SUFCMLA}}(\mathcal{A}_{mac}, \mathcal{L}_T, \mathcal{L}_V) \leq \mathbf{Adv}_{\mathcal{F}}^{\mathsf{LPRF}}(\mathcal{A}_{lprf}, \mathcal{L}_F) + \frac{q}{2^t - q}\,.$$

Note that the above theorem states that for any LPRF $\mathcal{F}$ the canonical MAC $\textsc{Mac}[\mathcal{F}]$ is SUF-CMLA-secure with the same message space as $\mathcal{F}$. In order to let the MAC handle arbitrarily long inputs, we need $\mathcal{F}$ to handle arbitrarily long inputs. This is achieved by first hashing the (arbitrarily long) input using a collision-resistant hash function and then applying the function $\mathcal{F}$. The resulting construction yields an LPRF with arbitrary input length as has been shown by Degabriele et al. [14, Theorem 5].

5.2 $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ is IND-CPLA Secure

The theorem below states that the encryption scheme $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ constructed from a fixed-input-length function $\mathcal{F}$ and a PRG $\mathcal{G}$ (cf. Fig. 7) is IND-CPLA-secure against semi-nonce-respecting adversaries if $\mathcal{F}$ is an LPRF and $\mathcal{G}$ is a secure PRG. We essentially show that the proof from [14] also holds for semi-nonce-respecting adversaries. Since we only need IND-CPLA security, we prove it for this case, adaptation to IND-aCPLA is straightforward. The proof is given in Appendix A.2.

Theorem 14 *Let* $\textsc{Se}[\mathcal{F}, \mathcal{G}]$ *be the mirror-like encryption scheme depicted in Fig. 7, composed of a fixed-input-length function family* $\mathcal{F}$ *and a PRG* $\mathcal{G}$ *with respective associated leakage sets* $\mathcal{L}_F$ *and* $\mathcal{L}_G$*. Then, for any semi-nonce-respecting* IND-CPLA *adversary* $\mathcal{A}_{se}$ *against* $\textsc{Se}[\mathcal{F}, \mathcal{G}]$*, making* q *queries to* Enc*, and associated leakage sets* $\mathcal{L}_E = \mathcal{L}_F \times \mathcal{L}_G$*, there exist an* LPRF *adversary* $\mathcal{A}_{lprf}$ *against* $\mathcal{F}$ *and a* PRG *adversary* $\mathcal{A}_{prg}$ *against* $\mathcal{G}$ *such that:*

$$\mathbf{Adv}^{\mathsf{INDCPLA}}_{\textsc{Se}[\mathcal{F},\mathcal{G}]}(\mathcal{A}_{se}, \mathcal{L}_E) \leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{lprf}, \mathcal{L}_F) + q\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{prg})\,.$$

The difference to [14] is that they consider PRGs which can be queried multiple times while we stick to the standard *single query* case. This entails that we need a hybrid argument over the q encryption queries by $\mathcal{A}_{se}$, which induces the factor q. In [14], the hybrid argument appears in the proof of the sponge-based PRG.

5.3 Security of the FGHF$'$ Construction

We can now state our main result, the following theorem, which states that the FGHF$'$ construction yields an LAE-secure AEAD scheme, if the underlying functions $\mathcal{F}$ and $\mathcal{F}'$ are leakage-resilient pseudorandom, $\mathcal{G}$ is a secure PRG, and $\mathcal{H}$ is a collision-resistant hash function. The theorem follows directly from Theorem 12, Theorem 13, and Theorem 14 combined with [14, Theorem 5]. The implications are also illustrated in Fig. 12.

Theorem 15 (LAE **Security of the** FGHF$'$ **Construction)** *Let* $\mathcal{F}$ *be a fixed-input-length* LPRF*,* $\mathcal{G}$ *a* PRG*,* $\mathcal{H}$ *a vector hash function, and* $\mathcal{F}'$ *be a fixed-input-length* LPRF *with associated leakage sets* $\mathcal{L}_F$*,* $\mathcal{L}_G$*,* $\mathcal{L}_H$*, and* $\mathcal{L}_{F'}$*, respectively. Let* FGHF$'$ *be the composition of* $\mathcal{F}$*,* $\mathcal{G}$*,* $\mathcal{H}$*, and* $\mathcal{F}'$ *(see Fig. 7) with associated leakage sets* $\mathcal{L}_{AE} = \mathcal{L}_{VD} = \mathcal{L}_F \times \mathcal{L}_G \times \mathcal{L}_H \times \mathcal{L}_{F'}$*. Then for any nonce-respecting* LAE *adversary* $\mathcal{A}_{ae}$ *against* FGHF$'$*, making* q_E *and* q_D *queries to* Enc *and* Dec*, respectively, there exist adversaries* $\mathcal{A}_{lprf}$*,* $\overline{\mathcal{A}}_{lprf}$*,* $\mathcal{A}_{prg}$*, and* $\mathcal{A}_{hash}$ *such that:*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{LAE}}_{\mathrm{FGHF}'}(\mathcal{A}_{ae}, \mathcal{L}_{AE}, \mathcal{L}_{VD}) \leq{}& \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{lprf}, \mathcal{L}_F) + 2\,\mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}'}(\overline{\mathcal{A}}_{lprf}, \mathcal{L}_{F'}) \\ &+ q_E\,\mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{A}_{prg}) + 2\,\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}_{hash}) + \frac{q_D}{2^t - q_D}\,.\end{aligned}$$

Theorem 15 improves [14, Theorem 6] by removing the additional requirement of unpredictability under leakage imposed on $\mathcal{F}'$. This entails that any instantiation of the FGHF$'$ construction can rely on the same function to instantiate $\mathcal{F}$ and $\mathcal{F}'$, thus one could name it FGHF instead. Indeed, the sponge-based instantiation $\textsc{Slae}$ [14] uses the same function to instantiate $\mathcal{F}$ and $\mathcal{F}'$, however, pseudorandomness and unpredictability under leakage were proven separately.

Acknowledgements

We thank Jean Paul Degabriele and Christian Janson for helpful discussions. This work was funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297.

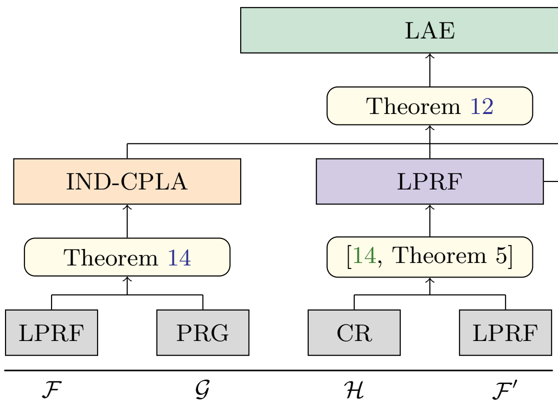

Fig. 12: Our security implications of the FGHF′ construction (cf. Theorem 15).

References

1. Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 471–488. Springer, Heidelberg, August 2013.
2. Farzaneh abed, Francesco Berti, and Stefan Lucks. Insecurity of RCB: Leakage-resilient authenticated encryption. Cryptology ePrint Archive, Report 2016/1121, 2016. http://eprint.iacr.org/2016/1121.
3. Megha Agrawal, Tarun Kumar Bansal, Donghoon Chang, Amit Kumar Chauhan, Seokhie Hong, Jinkeon Kang, and Somitra Kumar Sanadhya. Rcb: leakage-resilient authenticated encryption via re-keying. *The Journal of Supercomputing*, 74(9):4173–4198, Sep 2018.
4. Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2017.
5. 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, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 693–723. Springer, Heidelberg, December 2017.
6. Guy Barwell, Daniel Page, and Martijn Stam. Rogue decryption failures: Reconciling AE robustness notions. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *LNCS*, pages 94–111. Springer, Heidelberg, December 2015.
7. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, December 2000.
8. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.
9. Daniel J. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 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. *IACR Cryptology ePrint Archive*, 2019:137, 2019.
11. Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage-resilient and misuse-resistant authenticated encryption. Cryptology ePrint Archive, Report 2016/996, 2016. http://eprint.iacr.org/2016/996.
12. Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symm. Cryptol.*, 2017(3):271–293, 2017.
13. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Heidelberg, August 1999.
14. Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 209–240. Springer, Heidelberg, December 2019.
15. Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – towards side-channel secure authenticated encryption. *IACR Trans. Symm. Cryptol.*, 2017(1):80–105, 2017.
16. Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 21–40. Springer, Heidelberg, August 2010.

17. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th FOCS*, pages 293–302. IEEE Computer Society Press, October 2008.
18. Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 213–232. Springer, Heidelberg, September 2012.
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. Cryptology ePrint Archive, Report 2018/484, 2018. https://eprint.iacr.org/2018/484.
20. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT 2019*, volume 11774 of *LNCS*, pages 150–172. Springer, Heidelberg, 2019.
21. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards lightweight side-channel security and the leakage-resilience of the duplex sponge. *IACR Cryptology ePrint Archive*, 2019:193, 2019.
22. Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, and Michael Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 223–242. Springer, Heidelberg, December 2014.
23. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer, Heidelberg, February 2004.
24. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014.
25. National Institute of Standards and Technology. Lightweight cryptography standardization process, 2015.
26. 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, *ACM CCS 2015*, pages 96–108. ACM Press, October 2015.
27. Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, November 2002.
28. François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 335–352. Springer, Heidelberg, August 2013.
29. François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.
30. Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 223–238. Springer, Heidelberg, February / March 2013.
31. 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, *ACM CCS 2010*, pages 141–151. ACM Press, October 2010.

A Proofs for Section 5

A.1 Proof of Theorem 13

Proof. We prove the theorem using games G_0, G_1, and G_2 (cf. Fig. 13), where G_0 and G_2 are the game $\mathsf{SUFCMLA}$ with secret bit 1 and 0, respectively. Similar to (1), we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathsf{SUFCMLA}}_{\mathrm{MAC}[\mathcal{F}]}(\mathcal{A}_{mac}) &= 2\Pr\left[\mathsf{SUFCMLA}^{\mathcal{A}_{mac}} \Rightarrow \text{true}\right] - 1\\ &= \Pr\left[\mathcal{A}^{\mathsf{SUFCMLA}}_{mac} \Rightarrow 1 \,|\, b = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{SUFCMLA}}_{mac} \Rightarrow 1 \,|\, b = 0\right]\\ &= \Pr\left[\mathcal{A}^{\mathsf{G}_0}_{mac} \Rightarrow 1\right] + \Pr\left[\mathcal{A}^{\mathsf{G}_2}_{mac} \Rightarrow 1\right]\\ &= \sum_{i=1}^{2}\left(\Pr\left[\mathcal{A}^{\mathsf{G}_{i-1}}_{mac} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{G}_i}_{mac} \Rightarrow 1\right]\right). \end{aligned} \tag{8}$$

To bound the first term, we construct an LPRF adversary $\mathcal{A}_{lprf}$ against $\mathcal{F}$ as follows. It runs $\mathcal{A}_{mac}$ and answers all leakage queries (to LTag and LVfy) using its own leakage oracle LF. More formally, queries (X, L_t) to LTag are forwarded to LF as is the response (T, Λ_t) back to $\mathcal{A}_{mac}$. For queries (X, T, L_v) to LVfy, $\mathcal{A}_{lprf}$ obtains (T', Λ_v) by querying LF on (X, L_v). It sends (V, Λ_v) back to $\mathcal{A}_{mac}$, where $V \leftarrow \top$ if $T = T'$ and $V \leftarrow \perp$ otherwise. Whenever $\mathcal{A}_{mac}$ makes a query (X, T) to Vfy, $\mathcal{A}_{lprf}$ forwards X to F to obtain T'. If $T = T'$ it sends $\top$ to $\mathcal{A}_{mac}$, otherwise, it sends $\perp$.

It is easy to see that $\mathcal{A}_{lprf}$ perfectly simulates G_0 and G_1 for $\mathcal{A}_{mac}$ conditioned on the secret bit of game LPRF being 1 and 0, respectively. For every leakage (challenge) query by $\mathcal{A}_{mac}$, $\mathcal{A}_{lprf}$ makes exactly one leakage (challenge) query. As an SUF-CMLA adversary, $\mathcal{A}_{mac}$ does not forward queries to or from its challenge oracle, which yields that $\mathcal{A}_{lprf}$ does not make any prohibited query. Hence we conclude with

$$\Pr\left[\mathcal{A}^{\mathsf{G}_0}_{mac} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{G}_1}_{mac} \Rightarrow 1\right] \leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{lprf}, \mathcal{L}_F)\,. \tag{9}$$

For the second term, note that the leakage oracles LTag and LVfy are independent of the challenge oracle Vfy in both G_1 and G_2, thus we only consider the challenge oracle Vfy. To distinguish the games, the adversary $\mathcal{A}_{mac}$ has to make a query (X, T) to Vfy which would result in $\top$ in G_1. This means that $\mathcal{A}_{mac}$ has to guess the value $f[X]$ for an arbitrary X. Since $f[\,]$ is sampled randomly from $\{0,1\}^t$, we have

$$\Pr\left[\mathcal{A}^{\mathsf{G}_1}_{mac} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{G}_2}_{mac} \Rightarrow 1\right] \leq \frac{q}{2^t - q}\,. \tag{10}$$

Inserting (9) and (10) into (8) proves the claim. □

```
procedure Initialize
  $K \leftarrow_\$ \mathcal{K}$

procedure LTag$(X, L)$
  $\Lambda \leftarrow L(K, X)$
  $T \leftarrow \mathcal{F}(K, X)$
  return $(T, \Lambda)$

procedure LVfy$(X, T, L)$
  $\Lambda \leftarrow L(K, X)$
  $T' \leftarrow \mathcal{F}(K, X)$
  if $T' = T$
    return $(\top, \Lambda)$
  return $(\perp, \Lambda)$
```

```
procedure Vfy$(X, T)$ in $\mathsf{G}_0$
  $T' \leftarrow \mathcal{F}(K, X)$
  return $(T' = T)$

procedure Vfy$(X, T)$ in $\mathsf{G}_1$
  if $f[X] = \perp$
    $f[X] \leftarrow_\$ \{0,1\}^t$
  return $(f[X] = T)$

procedure Vfy$(X, T)$ in $\mathsf{G}_2$
  return $\perp$
```

Fig. 13: Games G_0, G_1, and G_2 used in the proof of Theorem 13. The leakage oracles LTag and LVfy are the same across the games. In each includes the challenge oracle Vfy as specified by the

A.2 Proof of Theorem 14

Proof. The proof works through a sequence of games $\mathsf{G}_0, \ldots, \mathsf{G}_3$ (cf. Fig 14). Game G_0 is the game $\mathsf{INDCPLA}$ instantiated with $\mathrm{SE}[\mathcal{F}, \mathcal{G}]$ and secret bit fixed to 1 and game G_3 is the same with secret bit fixed to 0. Using the equivalent notion of adversarial advantage, similar to (1), we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathsf{INDCPLA}}_{\mathrm{SE}[\mathcal{F},\mathcal{G}]}(\mathcal{A}_{se}, \mathcal{L}_E) &= 2\Pr\left[\mathsf{INDCPLA}^{\mathcal{A}_{se}} \Rightarrow \text{true}\right] - 1\\ &= \Pr\left[\mathcal{A}_{se}^{\mathsf{INDCPLA}} \Rightarrow 1 \,|\, b = 1\right] - \Pr\left[\mathcal{A}_{se}^{\mathsf{INDCPLA}} \Rightarrow 1 \,|\, b = 0\right]\\ &= \Pr\left[\mathcal{A}_{se}^{\mathsf{G}_0} \Rightarrow 1\right] + \Pr\left[\mathcal{A}_{se}^{\mathsf{G}_3} \Rightarrow 1\right]\\ &= \sum_{i=1}^{3}\left(\Pr\left[\mathcal{A}_{se}^{\mathsf{G}_{i-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{se}^{\mathsf{G}_i} \Rightarrow 1\right]\right) . \end{aligned} \tag{11}$$

We start with the first term for which we construct the following LPRF adversary $\mathcal{A}_{lprf}$ against $\mathcal{F}$. For queries (N, M) to Enc by $\mathcal{A}_{se}$, $\mathcal{A}_{lprf}$ invokes its own challenge oracle on N to obtain S. Subsequently, it computes $R \leftarrow \mathcal{G}(S, |M|)$ and sends $C \leftarrow R \oplus M$ to $\mathcal{A}_{se}$. Leakage queries $(N, M, (L_F, L_G))$ are processed as follows. The tuple (S, Λ_F) is obtained from the oracle LF upon querying it on (N, L_F), while $C \leftarrow \mathcal{G}(S, |M|) \oplus M$ and $\Lambda_G \leftarrow L_G(S, M)$ are computed locally by $\mathcal{A}_{lprf}$ before sending $(C, (\Lambda_F, \Lambda_G))$ to $\mathcal{A}_{se}$. Note that this proof is essentially the one from [14]. The difference is that we consider $\mathcal{A}_{se}$ to be semi-nonce-respecting ([14] restricts it to be nonce-respecting). This entails that $\mathcal{A}_{se}$ is allowed to make two queries $(N, M_1, (L_F, L_G))$ and $(N, M_2, (L_F, L_G))$ to its leakage oracle LEnc. By construction, $\mathcal{A}_{lprf}$ would invoke its leakage oracle LF twice on (N, L_F), which is not a problem since the game LPRF does not restrict such queries. Alternatively, $\mathcal{A}_{lprf}$ could keep a table of its queries and first look up whether it already made such a query and answer accordingly.

Since the games G_0 and G_1 solely differ in generating the seed S for $\mathcal{G}$ during queries to Enc, $\mathcal{A}_{lprf}$ perfectly simulates game G_0 and G_1 conditioned on the secret bit of the game LPRF being 1 and 0, respectively. We conclude with

$$\Pr[\mathcal{A}_{se}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathcal{A}_{se}^{\mathsf{G}_1} \Rightarrow 1] \leq \mathbf{Adv}^{\mathsf{LPRF}}_{\mathcal{F}}(\mathcal{A}_{lprf}, \mathcal{L}_F) . \tag{12}$$

Next, we bound the game hop between G_1 and G_2 for which we introduce a sequence of hybrid games $\mathsf{H}_0, \ldots, \mathsf{H}_q$, displayed in Fig. 15. The games differ in how the value R is sampled. In H_i, for the first i queries, it is sampled ideally, i.e., a random bit string of length identical to the queried message. For the remaining $q - i$ queries, it is sampled real, i.e., the output of the PRG $\mathcal{G}$ on input a random seed and the length of the queried message. It follows that $\mathsf{H}_0 = \mathsf{G}_1$ and $\mathsf{H}_q = \mathsf{G}_2$, hence

$$\begin{aligned}\Pr[\mathcal{A}_{se}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{se}^{\mathsf{G}_2} \Rightarrow 1] &\leq \Pr[\mathcal{A}_{se}^{\mathsf{H}_0} \Rightarrow 1] - \Pr[\mathcal{A}_{se}^{\mathsf{H}_q} \Rightarrow 1]\\ &\leq \sum_{i=1}^{q}\left(\Pr\left[\mathcal{A}_{se}^{\mathsf{H}_{i-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{se}^{\mathsf{H}_i} \Rightarrow 1\right]\right)\end{aligned}$$

We construct PRG adversaries $\mathcal{R}_1, \ldots, \mathcal{R}_q$ to bound the game hops between each pair of consecutive hybrid games. Adversary $\mathcal{R}_i$ proceeds as follows. It samples a key K for the function $\mathcal{F}$. This allows to perfectly simulate the leakage oracle LEnc for $\mathcal{A}_{se}$. For leakage queries $(N, M, (L_F, L_G))$, it (locally) computes $S \leftarrow \mathcal{F}(K, N)$, $C \leftarrow \mathcal{G}(S, |M|) \oplus M$, $\Lambda_F \leftarrow L_F(K, N)$, and $\Lambda_G \leftarrow L_G(S, M)$, and sends $(C, (\Lambda_F, \Lambda_G))$ to $\mathcal{A}_{se}$. Repeating nonces of the semi-nonce-respecting adversary $\mathcal{A}_{se}$ do not cause any issues, $\mathcal{R}_i$ can even keep a table of nonces with corresponding outputs and leakage to respond correctly. Let $(N_1, M_1), \ldots, (N_q, M_q)$ denote the challenge queries that $\mathcal{A}_{se}$ makes to Enc. For queries (N_j, M_j) with $j < i$, $\mathcal{R}_i$ samples $R \leftarrow\!\!\$\, \{0,1\}^{|M_j|}$ and sends $C \leftarrow R \oplus M_j$ back to $\mathcal{A}_{se}$. For queries (N_j, M_j) with $j > i$, $\mathcal{R}_i$ samples a seed S for $\mathcal{G}$, computes $R \leftarrow \mathcal{G}(S, |M_j|)$ and sends $C \leftarrow R \oplus M_j$ back to $\mathcal{A}_{se}$. For the i-th query, (N_i, M_i), $\mathcal{R}_i$ invokes its own oracle G on $|M_i|$ to obtain R, computes $C \leftarrow R \oplus M_i$. It is easy to see that $\mathcal{R}_i$ simulates H_{i-1} and H_i if its challenge bit b from the game PRG is 1 and 0, respectively. Hence, we have

$$\Pr\left[\mathcal{A}_{se}^{\mathsf{H}_{i-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{se}^{\mathsf{H}_i} \Rightarrow 1\right] \leq \mathbf{Adv}^{\mathsf{PRG}}_{\mathcal{G}}(\mathcal{R}_i) .$$

Collecting everything and defining $\mathcal{A}_{prg}$ to be the adversary with the highest PRG advantage among $\mathcal{R}_1, \ldots, \mathcal{R}_q$ yields

$$\begin{aligned}
\Pr[\mathcal{A}_{se}^{\mathsf{G}_1} \Rightarrow 1] - \Pr[\mathcal{A}_{se}^{\mathsf{G}_2} \Rightarrow 1] &\leq \sum_{i=1}^{q} \left(\Pr\left[\mathcal{A}_{se}^{\mathsf{H}_{i-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{se}^{\mathsf{H}_i} \Rightarrow 1\right]\right) \\
&\leq \sum_{i=1}^{q} \mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{R}_i) \\
&\leq \sum_{i=1}^{q} \mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{A}_{prg}) \\
&\leq q\mathbf{Adv}_{\mathcal{G}}^{\mathsf{PRG}}(\mathcal{A}_{prg})\,. \qquad (13)
\end{aligned}$$

Since the challenge oracles Enc in G_2 and G_3 output identically distributed ciphertexts the adversary can not distinguish these games which yields

$$\Pr[\mathcal{A}_{se}^{\mathsf{G}_2} \Rightarrow 1] - \Pr[\mathcal{A}_{se}^{\mathsf{G}_3} \Rightarrow 1] = 0\,. \qquad (14)$$

Inserting (12), (13), and (14) into (11) yields the desired result. □

```
procedure Initialize
  $K \leftarrow_{\$} \mathcal{K}$

procedure Enc$(N, M)$ in $\mathsf{G}_0$ and $\boxed{\mathsf{G}_1}$
  $S \leftarrow \mathcal{F}(K, N)$
  $\boxed{S \leftarrow_{\$} \{0,1\}^n}$
  $R \leftarrow \mathcal{G}(S, |M|)$
  return $C \leftarrow R \oplus M$

procedure LEnc$(N, M, (L_F, L_G))$
  $S \leftarrow \mathcal{F}(K, N)$
  $R \leftarrow \mathcal{G}(S, |M|)$
  $C \leftarrow R \oplus M$
  $\Lambda_F \leftarrow L_F(K, N)$
  $\Lambda_G \leftarrow L_G(S, M)$
  return $(C, (\Lambda_F, \Lambda_G))$
```

```
procedure Enc$(N, M)$ in $\mathsf{G}_2$
  $R \leftarrow_{\$} \{0,1\}^{|M|}$
  return $C \leftarrow R \oplus M$

procedure Enc$(N, M)$ in $\mathsf{G}_3$
  return $C \leftarrow_{\$} \{0,1\}^{|M|}$
```

Fig. 14: Games G_0, G_1, G_2, and G_3 used to prove Theorem 14. Game G_1 contains the boxed code, G_0 does not. For games G_2 and G_3 we only display the oracle Enc as the leakage oracles are identical to those in G_0 and G_1.

```
procedure Initialize
  $K \leftarrow_{\$} \mathcal{K}$;  $c \leftarrow 0$

procedure LEnc$(N, M, (L_F, L_G))$
  $S \leftarrow \mathcal{F}(K, N)$
  $R \leftarrow \mathcal{G}(S, |M|)$
  $C \leftarrow R \oplus M$
  $\Lambda_F \leftarrow L_F(K, N)$
  $\Lambda_G \leftarrow L_G(S, M)$
  return $(C, (\Lambda_F, \Lambda_G))$

procedure Enc$(N, M)$
  $c \leftarrow c + 1$
  if $c \leq i$
    $R \leftarrow_{\$} \{0,1\}^{|M|}$
  else
    $S \leftarrow_{\$} \{0,1\}^n$
    $R \leftarrow \mathcal{G}(S, |M|)$
  return $C \leftarrow R \oplus M$
```

Fig. 15: Hybrid games H_i used to prove Theorem 14.