Authenticated Encryption in the Face of Protocol and Side Channel Leakage

Guy Barwell[1], Daniel P. Martin[2], Elisabeth Oswald[1], and Martijn Stam[1]

[1] Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol, BS8 1UB, United Kingdom.
{guy.barwell,elisabeth.oswald, martijn.stam}@bris.ac.uk
[2] Heilbronn Institute for Mathematical Research, University of Bristol,
Howard House, Queen's Avenue,
Bristol, United Kingdom**.
dan.martin@bris.ac.uk

Abstract. Authenticated encryption schemes in practice have to be robust against adversaries that have access to various types of leakage, for instance decryption leakage on invalid ciphertext (protocol leakage), or leakage on the underlying primitives (side channel leakage). Our work includes several novel contributions: we augment the notion of nonce-base authenticated encryption with the notion of continuous leakage and we prove composition results in the face of protocol and side channel leakage. Moreover, we show how to achieve authenticated encryption that is simultaneously both misuse resistant and leakage resilient, based on a sufficiently leakage resilient PRF, and finally we propose a concrete, pairing-based instantiation of the latter.

Keywords: provable security, authenticated encryption, generic composition, leakage resilience, robustness

** This research was carried out while D. P. Martin was a member of the Department of Computer Science, University of Bristol

Table of Contents

1 Introduction . . . 3
  1.1 Our contributions . . . 3
  1.2 Structure . . . 5
  1.3 Related work . . . 5
2 Preliminaries . . . 6
  2.1 Authenticated Encryption . . . 7
3 Security Notions in the Presence of Leakage . . . 10
  3.1 Implementations versus Functions . . . 11
  3.2 What Constitutes Leakage . . . 12
  3.3 Security Notions including Leakage . . . 12
4 Applying LAE to attacks in theory and practice . . . 15
  4.1 Theoretical leakage models . . . 15
  4.2 A practical example from the literature . . . 16
5 Generic Composition for LAE . . . 17
  5.1 Modelling Composed Leakage . . . 17
  5.2 MAC-and/then-Encrypt are Brittle under Leakage . . . 18
  5.3 Encrypt-then-MAC is Secure under Leakage . . . 19
  5.4 Achieving Misuse Resistant LAE security . . . 20
6 A mrLAE scheme . . . 21
7 Conclusions and Open Problems . . . 22
References . . . 23
A Security Proof for Generic Composition . . . 25
B An instantiation of the SIVAT mechanism . . . 28
  B.1 Recalling the MOSW MAC . . . 29
  B.2 Modelling Discussion . . . 30
  B.3 A Leakage-resiliant PRF . . . 30
  B.4 A fully Leakage-resilient MAC . . . 32
  B.5 A Leakage-resiliant IV-based Encryption Scheme . . . 33
  B.6 nLAE and mrLAE constructions in the generic group model . . . 34
C Security Proofs for Leakage Resilient Components . . . 34

1 Introduction

Authenticated Encryption (AE) has arisen out of (practical) necessity: historic modes-of-operation for symmetric encryption [40] implicitly target confidentiality against passive adversaries. However, in most realistic threat models security against active adversaries is desired as well. Thwarting adversaries trying to modify ciphertexts is best captured by requiring ciphertext integrity; encryption schemes that offer both this and a suitable passive indistinguishability notion are said to provide authenticated encryption. Today, authenticated encryption has become the primitive of choice to enable secure communication. AE schemes can be constructed from components that individually provide either confidentiality or authenticity, both in a traditional probabilistic setting [8] and a more modern nonce-based one [38]. As a result, there exist several black-box constructions of authenticated encryption schemes based on simpler, keyed primitives such as pseudorandom functions or permutations, including MACs and blockciphers.

Unfortunately, in practice neither the composition nor the underlying components behave as black-boxes: side-channel attacks often leak additional information to an adversary, leading to real-life breaks [56]. Invariably, these attacks are possible by exploiting a discrepancy between the capabilities of a theoretical adversary and an actual, real-life one. Thus, these attacks neither violate the security assumptions on the primitive nor do they invalidate the security claims: rather, they render these claims insufficient and the existing security models as inadequate.

In response, a number of works have tried to capture more closely how protocols behave when implemented [13, 19, 23]. We are particularly interested in *subtle* authenticated encryption [4] which augments the authenticated encryption security game with an implementation-dependent leakage oracle that provides an adversary deterministic decryption leakage on *invalid* ciphertexts only. Subtle authenticated encryption encompasses earlier notions such as multiple decryption errors [12] and the release of unverified plaintexts [2]; it can be regarded as *protocol* leakage.

Orthogonally, *primitives* can leak. Kocher (*et al.*) [28, 29] showed how both timing and power measurements lead to a side-channel, enabling the extraction of secret data out of cryptographic devices. Primitives believed to be secure, such as AES, were broken without actually violating the assumption that AES is a secure pseudorandom permutation. Such attacks are captured in the framework of leakage resilient cryptography. Here an adversary can adaptively choose a leakage function that is restricted in scope as only computation is assumed to leak information [37], and in size. The latter is captured by leaking only a certain number of bits per call. If the overall leakage remains unbounded the model is referred to as continuous leakage. For a variety of schemes and security notions, resilience against certain classes of leakage can be proven [15, 27, 54], but dealing with adaptivity that allows leakage after an adversary has received a challenge is often problematic.

The current theory of authenticated encryption is not suited to take this additional leakage resource into account. In this work we provide a framework for dealing with AE in the presence of leakage, which then allows us to determine the constraints on primitives and constructions alike to yield AE secure against classes of leakage functions. Moreover, we propose a concrete instantiation of a leakage-resilient pseudorandom function suitable to be used to form the first leakage-resilient, nonce-based authenticated encryption scheme.

1.1 Our contributions

Augmenting nonce-base authenticated encryption with leakage. We start with augmenting the nonce-based authenticated encryption security notion (Section 2.1) with leakage (Section 3). This new notion, which we will refer to as LAE, can be regarded as a generalization of the SAE framework by Barwell *et al.* [4], yet it also captures leakage-resilience as introduced by Dziembowski and Pietrzak [17]. We provide corresponding leakage notions for the primitives used by the composition results by Namprempre *et al.* [38] (NRS), namely nonce- or iv-based encryption, pseudorandom functions, and message authentication codes.

For the traditional AE notion by Rogaway and Shrimpton [49], an adversary has to distinguish between a world with a real encryption and decryption oracle on the one hand, and a world with a random ciphertext generator and a rejection oracle on the other. In the LAE game the number of oracles available to the adversary increased from two to four: both worlds are augmented with true encryption and decryption oracles and we will allow (only) these additional oracles to leak.

For the leakage mechanism, we adopt the approach originally suggested by Micali and Reyzin [37] and later adapted for leakage resilience [17] where an adversary can provide a leakage function to be evaluated on the internal variables of the oracle, with the leakage output to be returned to the adversary alongside the normal output. The model is very powerful, allowing the adversary to adaptively choose which leakage function they would like evaluated on a query by query basis.

To avoid trivial wins, the leakage functions that are allowed need to be restricted, to prevent, for instance, leaking the entire key in one go. We model this by explicitly defining security relative to a class of leakage functions (as is common for instance in the contexts for related key or key-dependent message attacks). By appropriately setting the class of leakage functions, we show that our notion generalises previous strengthened AE security notions, including SAE, RUP and distinguishable decryption errors [2, 4, 12], and previous leakage notions, including the simulatable leakage, auxiliary input and probing models [15, 24, 54].

Generic composition with leakage. Our second contribution (Section 5) is an investigation on how to perform generic composition in the presence of leakage by extending the results of Namprempre *et al.* [38] (henceforth NRS). We establish that schemes susceptible to release of unverified plaintext are unsuitable even for much more modest types of leakage and we confirm modern folklore that this affects all schemes that are roughly of the type Encrypt-and-MAC or MAC-then-Encrypt (cf. [2]). Conversely, we show that Encrypt-then-MAC style schemes *are* secure against a large class of leakage functions, where we express this class in terms of the leakage classes against which the underlying primitives are secure. For this composition of leakage from different primitives, we effectively just concatenate the leakage of the constituent parts, which implicitly assumes that only computation leaks (cf. [37]).

In particular, we show security of the N2 and A5 constructions of NRS against nonce-respecting adversaries (Theorems 1 and 3), and of A6 against adversaries who never repeat a nonce and associated-data pair (Theorem 4).

While the above result shows that none of the NRS schemes achieve misuse resistant LAE security, we go on to give a generic construction that does meet this strongest definition of security, albeit at the cost of further ciphertext expansion (Theorem 5). Our result gives ciphertexts that are two blocks longer than the messages, we leave open whether mrlAE security can be achieved with less ciphertext expansion.

Moreover, we show that instantiating CFB mode with a pseudorandom function yields a secure iv-based encryption scheme even under leakage (Theorem 10). This allows us to apply our generic composition results to construct the first AE scheme secure against continuous leakage based on a pseudorandom function actively secure against continuous leakage and a MAC scheme secure against continuous leakage of both tagging and verification.

Instantiation using a new leakage resilient PRF. Our final contribution (Appendix B) is the construction of these latter two primitives. To this end, we extend the MAC of Martin *et al.* [34] in two directions. First, we show how it can be adapted such that it may leak under verification (Theorem 9) answering an open question from their work. While the MAC is a pseudorandom function when no leakage is present, already small amounts of leakage are disastrous for the pseudorandomness property. It turns out that the underlying key update mechanism due to Kiltz and Pietrzak [27] is intrinsically unsuitable to create an actively secure pseudorandom function: the mechanism shares a key out in two which allows a form of leak-in-the-middle attack. The solution we propose is to use three shares instead and we prove that the resulting construction is indeed a pseudorandom function that is leakage-resilient even against adaptive adversaries.

1.2 Structure

In Section 2 we give our notation, and recall a host of security definitions without leakage. The reason for the depth within this section is due to our definitions having a slightly different representation from what is "standard". Taking AE as an example, the adversary will have access to honest encryption and decryption oracles, as well as challenge oracles for both functionalities. It is fairly straightforward to show that the two definitions are equivalent. However, the more expressive definition will later become important, when we will enhance the honest oracle to also provide leakage. If the reader is familiar with these notions, the section may be skipped and referred back to as required.

Section 3 provides security definitions when the adversary is also allowed leakage from the computation of a function. Before we can do this we formalise the distinction between a function and its implementation. When leakage is involved we will discuss the security of a given implementation. An implementation will be shown secure against particular classes of leakage functions which are also discussed here.

Section 4 demonstrates how our new leakage resilient framework using classes of leakage functions captures a whole host of previous security models which involve some form of leakage. We also express some recent attacks in terms of our model.

Section 5 considers general composition, of leakage resilient primitives, to construct leakage resilient authenticated encryption. We consider the MAC-then-Encrypt, MAC-and-Encrypt and Encrypt-then-MAC paradigms and demonstrate which compositions maintain security when leakage is allowed. The section is concluded with a new generic composition which allows the construction of an authenticated encryption scheme which is both leakage resilient and misuse resistant. An instantiation of the scheme is touched upon, with the details being given in Appendix B.

1.3 Related work

Authenticated encryption. One of the earliest symmetric works on concrete security of AE was by Bellare and Namprempre [8]. Working within the probabilistic model, they formalised what it meant to be both confidential and authentic, and investigated how one could achieve this through generic composition, combining two schemes (one with each security property) such that their composition achieved both. Yet, modern authenticated encryption is a stateless and deterministic notion, taking in any randomness or state as an extra parameter termed the nonce. It was formalised across a number of papers, culminating in Rogaway and Shrimpton's 2006 work on DAE [49] and only recently a comprehensive study of all the ways one could combine a PRF with an encryption scheme was completed in the nonce-based setting [38].

The CAESAR competition [10] has driven further research into AE, and particularly into the concept of robustness, namely the idea that a scheme should be more resistant to common problems faced in the real-world. One branch of this research has been into designing schemes that are resistant to certain forms of leakage. Prior to the competition, Boldyreva *et al.* [12] had investigated how to model a scheme from which decryption failures are not identical, such as under a timing attack. Andreeva *et al.* [2] (RUP) considered the release of unverified plaintexts, where the decryption oracle releases candidate plaintexts even if they fail verification. The robust authenticated encryption notion of Hoang *et al.* [23] also implies security against the leakage of these candidate plaintexts, among other goals. Barwell *et al.* [4] defined the SAE framework as a generalisation of these notions, and used it to compare the three previous works. However, in each of these cases the adversary only receives leakage from decryption, and this leakage is modelled as a fixed, deterministic function, rather than a more general set of functions available to an adaptive side-channel attacker.

Leakage resilient constructions. Within the leakage resilient literature, there are several works towards providing leakage resilient encryption, but most of them have been in the bounded leakage model [22, 44]. In the bounded retrieval model, Bellare *et al.* [7] proved the security of a symmetric encryption scheme that provides authenticated encryption in the leak free case, and indistinguishability when leakage is involved. Pereira *et al.* [41] proposed what is, to our knowledge, the first and only leakage resilient encryption scheme in the simulatable leakage model. However, the construction requires a leak free component, and work by Longo *et al.* [31] shows that there are currently no efficient simulators.

Another manner to ensure that the adversary can not progressively leak the key material is to update the keys themselves (instead of their representation). Previous leakage resilient works in this direction include the MAC of Schipper [51], or the DH-ratcheting concept (*e.g.* [42]). However, these tend to require that all parties to the communication hold modifiable state and remain perfectly in sync, a demand we are able to avoid.

Each of these existing models severely restricts the information or computations that an adversary may be able to perform, limiting their utility for modelling active side-channel attacks, problems mitigated by the continuous leakage model, on which we will focus. To our knowledge there are no leakage resilient encryption schemes in the continuous leakage model.

Our generic composition results allow us to combine leakage resilient components, for which we provide candidates built around a PRF secure against leakage. Currently there are two leakage resilient PRGs, due to Pietrzak (and Dziembowski) [17, 43], from which it may be possible to build a leakage resilient stream cipher, although issues arise with restarting using the same key. Works of Dodis and Pietrzak [16], and Faust *et al.* [18] describe two PRFs secure under leakage. However, each requires that the leakage (and inputs) be fixed before the start of the game. For a PRF to be used within a composition theorem, adaptive security is required. Finally, Martin *et al.* [34] provide a MAC which is secure against leakage on the tagging function only. We will use this as the basis of our instantiations, and extend it to achieve security against leakage on verification queries, resolving an open question from their work.

2 Preliminaries

General notation. For assignment of a value U to the variable T we will write $T \leftarrow U$, where U may also be the outcome of some computation. If the variable is a set, we use the shorthand $S \xleftarrow{\cup} U$ for $S \leftarrow S \cup \{U\}$. To assign a value drawn uniformly at random from some finite set B to variable A, we write $A \leftarrow\!\!\$\ B$. By convention, arrays and lists are initialised empty. We use $=$ for equality testing. We write $\mathbb{A} \to b$, to denote that adversary $\mathbb{A}$ outputs some value b. To define notions etc. we will write $X := Y$ to say that X is defined as some expression Y. The distinguished symbol $\lightning$ denotes an invalid query. The symbol $||$ denotes an *unambiguous* encoding, meaning if $Z \leftarrow X||Y$ it must be possible given Z to uniquely recover X and Y, notated $X||Y \leftarrow Z$, no matter what types X, Y may take. The length $|A|$ is the length of A when expressed as a string of elements of some underlying alphabet Σ (usually $\Sigma = \{0, 1\}$).

Whenever a function is described with a subscript, this will define the first parameter, meaning $f_k(\cdot,\cdot) = f(k,\cdot,\cdot)$. For consistency and clarity of notation, we refer to security definitions in capitals (*e.g.* IND–CPA) and typeset functions in calligraphic ($\mathcal{E}$), spaces in sans serif (K), "secret" elements in lower case (k), known elements in upper case (M), and adversaries in blackboard bold ($\mathbb{A}$). When we introduce implementations, these will be denoted in bold ($\boldsymbol{\mathcal{E}}$).

Adversarial advantages. We will define our security notions through indistinguishability games where an adversary is given access to one of two collections of oracles. The adversary $\mathbb{A}$ may make queries to these oracles, and eventually outputs a bit. Instead of writing the games in code, we adopt shorthand notation [2] so that the *distinguishing advantage* of $\mathbb{A}$ between two collections of n oracles $(\mathcal{O}_1, \ldots, \mathcal{O}_n)$ and $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ is defined as

$$\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathcal{O}_1, \ldots, \mathcal{O}_n \\ \mathcal{P}_1, \ldots, \mathcal{P}_n\end{pmatrix} := \left|\Pr\left[\mathbb{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_n} \to 1\right] - \Pr\left[\mathbb{A}^{\mathcal{P}_1,\ldots,\mathcal{P}_n} \to 1\right]\right|,$$

where the probabilities are taken over the randomness of the oracles, and key $k \leftarrow\!\!\$\ \mathsf{K}$ (note that multiple oracles will often use the same key). We may refer to the oracles by their numerical position: the i^{th} oracle implements either $\mathcal{O}_i$ or $\mathcal{P}_i$ depending which collection the adversary is interacting with.

A scheme is considered secure with respect to a particular security goal if the relevant adversarial advantage is small for all adversaries running within reasonable resources. We do not draw judgement as to what "small" may mean, nor what constitutes "reasonable resources", since these as heavily dependent on context.

2.1 Authenticated Encryption

Core definitions. Early works on defining what entails symmetric encryption (cf. [25]) closely followed the earlier formalisms for public key encryption. Over the years understanding of what should be expected of symmetric encryption evolved considerably, both in terms of syntax and security. The basis for our work will be the widely accepted nonce-based model using indistinguishability from random bits for confidentiality [46–48]. After introducing this model, we will briefly refer back to an older, non-authenticated version of encryption as it is one of the building blocks later on.

Syntax. An authenticated encryption scheme consists of a pair of deterministic functions Enc and Dec, called encryption and decryption, respectively. Encryption Enc takes four inputs, resulting in a single ciphertext $C \in \mathsf{C}$. Besides the key $k \in \mathsf{K}$ and the message $M \in \mathsf{M}$, the inputs are some associated data $A \in \mathsf{A}$ that will be authenticated but not encrypted, and finally a nonce $N \in \mathsf{N}$ that will be used to ensure that repeat encryptions will not result in repeat ciphertexts. Decryption Dec takes as input again the key, the nonce, and the associated date, in addition to the ciphertext. It outputs a purported message or an error message $\perp \notin \mathsf{M}$.

This syntax can be summarized as

$$\begin{aligned}\mathsf{Enc}\colon \mathsf{K}\times\mathsf{N}\times\mathsf{A}\times\mathsf{M} &\rightarrow \mathsf{C}\\ \mathsf{Dec}\colon \mathsf{K}\times\mathsf{N}\times\mathsf{A}\times\mathsf{C} &\rightarrow \mathsf{M}\cup\{\perp\}\ .\end{aligned}$$

In practice, the key space K, nonce space N, associated data A, message space M, and ciphertext space C are bitstrings of various lengths. It is common to have $\mathsf{A}=\mathsf{M}=\mathsf{C}=\{0,1\}^*$, and $\mathsf{K}=\mathsf{N}=\{0,1\}^n$ for some security parameter n. That said, our implementation in Section B is given with respect to more general groups (linked to pairings) to instantiate the various spaces.

We require that an authenticated encryption scheme is both *correct* and *tidy*. These two properties are satisfied if, for all k, N, A, M, C in the appropriate spaces:

Correctness : $\mathsf{Dec}(k,N,A,\mathsf{Enc}(k,N,A,M)) = M$

Tidiness : if $\mathsf{Dec}(k,N,A,C) \neq \perp$ then $\mathsf{Enc}(k,N,A,\mathsf{Dec}(k,N,A,C)) = C$

Together, tidiness and correctness imply that decryption is wholly specified by the encryption routine.

Additionally, we require encryption to be *length regular*, which is satisfied if there exists some stretch function $\tau\colon \mathbb{N}\rightarrow\mathbb{N}$ such that for all inputs $|\mathsf{Enc}(k,N,A,M)| = |M| + \tau(|M|)$.

Security notions. Ever since Rogaway and Shrimpton's treatment of deterministic authenticated encryption, it is customary to capture both confidentiality and integrity requirements in a single game. Here the adversary gets oracle access either to the "real" world or to the "ideal" world and needs to distinguish between these two worlds. In the real world, oracle access consists of the encryption and decryption functionalities Enc_k and Dec_k, using a randomly drawn and secret key k. In the ideal world, the encryption oracle is replaced with an oracle \$ that generates randomly drawn ciphertexts and the decryption oracle with an oracle $\perp$ that rejects all ciphertexts. Irrespective of which world the adversary is in, we will refer to the Enc_k vs. \$ one as the challenge encryption oracle, (it will sometimes be referred to as the first oracle based on the oracle ordering) and to the Dec_k vs. $\perp$ one as the challenge decryption oracle.

We will use a slightly different, but equivalent, formulation where an adversary additionally has access to the true encryption and decryption oracles in both worlds. Thus the adversary will have access to *four* oracles in each world: the challenge encryption oracle, the challenge decryption oracle, the true encryption oracle, and finally the true decryption oracle. Having these extra oracles will help us later on to add leakage, which will only ever be on the true oracles and never on one of the challenge oracles. One

```
function $\$^F(X)$
    $C_0 \leftarrow F(X)$
    $C_1 \leftarrow\!\!\$\ \Sigma^{|C_0|}$
    return $C_1$

function $\bot^G(X)$
    return $\bot$
```

Fig. 1: The generic oracles $\$^F$ and $\bot^G$ idealise the output of F as random bits, and of G as always rejecting. They are used to define the reference world in our security definitions, for various choices of (F, G), which will be omitted whenever clear. Usually $\Sigma = \{0, 1\}$, with $|C_0|$ the length of C_0 as a bitstring.

could even argue that the additional oracles provide a more representative and expressive framework: the honest oracles describe how an adversary may "learn" about a system, while the challenge ones allow them to "prove" they have done so (cf. a similar, more detailed argument for subtle authenticated encryption [4]).

As our reference point we will use the oracles defined in Figure 1, with all probabilities taken over randomness of the key and sampling within the oracle.

Queries. Already in the leak-free setting, certain combinations of queries will easily distinguish the two worlds. To avoid these trivial wins, we will therefore prohibit certain queries—or in some cases simply assume adversaries refrain from making prohibited queries. For example, if an adversary can send a challenge encryption to decryption they can trivially win. As a general rule, we prohibit the same query being made to oracles which take the same inputs (such as the honest and challenge encryption oracles), and also prohibit performing the inverse of previous queries. For example, the ciphertext output from the either oracle cannot be passed into the decryption oracle.

If a query has already been made, we refer to the process of later making an equivalent query as *forwarding* the query. To *forward* a query is to make a second query who's inputs were inputs or outputs from the first query. Thus if an adversary has made a query (N, A, M) to an authenticated encryption oracle Enc to receive output C, it would be forwarding this to later make a query (N, A, M) to any oracle with syntax $\mathsf{N} \times \mathsf{A} \times \mathsf{M}$, or (N, A, C) to one with syntax $\mathsf{N} \times \mathsf{A} \times \mathsf{C}$, such as Dec. Making the same query again, *repeating* a query, can be thought as a special case of forwarding a query. When we later introduce leakage, we will ignore this for the purposes of defining forwarding of queries, so in the previous example the query (N, A, M, L) would also be considered forwarding.

Nonce selection requirements. Our security games will be agnostic over how the nonce is selected, with this property enforced by restricting the adversary. An adversary against an (authenticated) encryption scheme is called *nonce respecting* if whenever making a new query they do not use a nonce more than once to any oracle matching the syntax of Enc_k or $\mathcal{E}_k$. They are *random-iv respecting*, or simply *iv respecting*, if for any new query with these oracles their nonce N (which we term an IV and will generally write as I instead) is sampled uniformly from N immediately prior to querying the oracle (and thus not involved in the logic used to select other elements of the query). These requirements do not apply when interacting with oracles matching the syntax of Dec_k or $\mathcal{D}_k$. A scheme is called *(nonce) misuse resistant* if the adversary does not have to be nonce respecting, providing that the adversary does not make multiple queries using the same (N, A, M) triple.

Definition 1 (nAE). *Let* Enc *be an authenticated encryption scheme,* $\mathbb{A}$ *an adversary who does not make forward queries to or from his first or second oracle (and thus does not repeat first oracle queries). Then, the nAE advantage of an adversary* $\mathbb{A}$ *against* Enc *is*

$$\mathbf{Adv}^{\mathrm{AE}}_{\mathsf{Enc}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta} \begin{pmatrix} \mathsf{Enc}_k, \mathsf{Dec}_k, \mathsf{Enc}_k, \mathsf{Dec}_k \\ \$ \quad , \quad \bot \quad , \mathsf{Enc}_k, \mathsf{Dec}_k \end{pmatrix} .$$

It is a secure nAE *scheme (or simply nAE) if this advantage is small for all nonce-respecting adversaries running within reasonable resources, and mrAE if it is small for all adversaries running within reasonable resources.*

To aid clarity we offer an explanation of the definition of AE. The distinguishing advantage is taken over two sets of oracles, $(\mathsf{Enc}_k, \mathsf{Dec}_k, \mathsf{Enc}_k, \mathsf{Dec}_k)$ and $(\$, \perp, \mathsf{Enc}_k, \mathsf{Dec}_k)$, where the six keyed oracles use the same key. The restriction that an adversary may not make repeat queries or 'forward' a query (or its inverse) between oracles ensures that there are no trivial wins for the adversary.

Encryption. Authenticated encryption schemes are often constructed based on simpler encryption schemes. From a security perspective, the latter typically only provide a form of passive confidentiality and not integrity or authenticity. Consequently, these simpler encryption schemes do not take associated data as input, leading to the following syntax:

$$\mathcal{E}\colon \mathsf{K} \times \mathsf{N} \times \mathsf{M} \to \mathsf{C}$$
$$\mathcal{D}\colon \mathsf{K} \times \mathsf{N} \times \mathsf{C} \to \mathsf{M} \cup \{\perp\}$$

where $\mathcal{E}$ is length-regular and $\mathcal{D}$ is uniquely determined by $\mathcal{E}$ to ensure both correctness and tidiness.

The standard confidentiality notion for an encryption scheme is indistinguishability under chosen plaintext attacks (Def. 2). For iv-based, or ivE, schemes, only iv-respecting adversaries are considered, whereas for nonce-based, or nE, schemes only nonce-respecting adversaries are considered.

Definition 2 (Indistinguishability under chosen plaintext attacks). *Let $\mathcal{E}$ be an encryption scheme, and $\mathbb{A}$ an adversary who does not repeat first oracle queries or forward queries between his oracles. Then, the* IND–CPA *advantage of $\mathbb{A}$ against $\mathcal{E}$ is*

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{IND-CPA}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathcal{E}_k, \mathcal{E}_k \\ \$\,, \mathcal{E}_k\end{pmatrix}.$$

The scheme $\mathcal{E}$ is nE *(resp.* ivE*) if the IND–CPA advantage is small for all nonce-respecting (resp. iv-respecting) adversaries running within reasonable resources.*

Building blocks: MACs and PRFs. A Message Authentication Code (MAC) consists of a pair of deterministic functions $(\mathcal{T}, \mathcal{V})$, named for Tag and Verify, having the syntax:

$$\mathcal{T}\colon \mathsf{K} \times \mathsf{M} \to \mathsf{T}$$
$$\mathcal{V}\colon \mathsf{K} \times \mathsf{M} \times \mathsf{T} \to \{\top, \perp\}$$

where T is called the tag space.

We restrict ourselves to deterministic MACs since our overall objective is to define a scheme that can be implemented in a deterministic way, and hence each component must be deterministic. In this setting, it is common to omit $\mathcal{V}$, since recomputating and comparing this with the candidate tag suffices (similar to how as encryption defines decryption for any tidy encryption schemes). We opt for the more descriptive setting and make $\mathcal{V}$ explicit, since later when we introduce leakage it will be important to differentiate between the tagging and verification implementations. Indeed, when leakage is available, simply recomputing the MAC is no longer sufficient to achieve security.

Definition 3 (Strong Existential Unforgeability under Chosen Message Attack). *Let $(\mathcal{T}, \mathcal{V})$ be a (deterministic) MAC, and $\mathbb{A}$ an adversary who does not forward queries from his second oracle to the first. The Strong Existential Unforgeability under Chosen Message Attack (*sEUF-CMA*) advantage of $\mathbb{A}$ against $(\mathcal{T}, \mathcal{V})$ is the probability $\mathbb{A}$ can distinguish between interacting with $\mathcal{V}_k$ and $\perp$ when given access to $\mathcal{T}_k$ and $\mathcal{V}_k$. That is,*

$$\mathbf{Adv}_{(\mathcal{T},\mathcal{V})}^{\mathrm{sEUF-CMA}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathcal{V}_k, \mathcal{T}_k, \mathcal{V}_k \\ \perp\,, \mathcal{T}_k, \mathcal{V}_k\end{pmatrix}.$$

The pair $(\mathcal{T}, \mathcal{V})$ is a secure MAC if this advantage is small for all reasonably resourced adversaries.

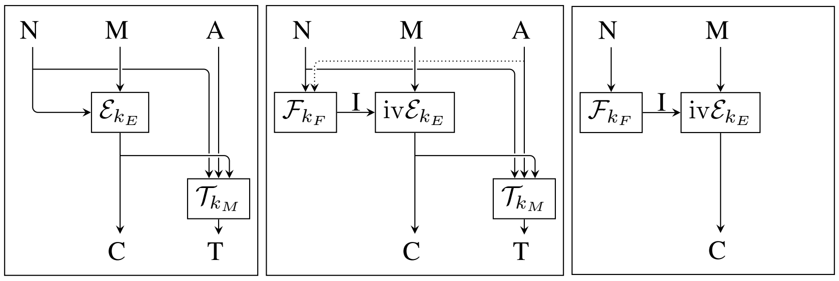

Fig. 2: Graphical representations of the encryption directions of generic composition mechanisms. On the left, N2 converts a nonce-based encryption algorithm $\mathcal{E}$ and MAC scheme $(\mathcal{T},\mathcal{V})$ into an nAE scheme. On the right, iv2n converts an iv-based encryption scheme iv$\mathcal{E}$ and a PRF into a nonce-based encryption algorithm. Composing these yields A5, shown in the middle ignoring the dotted input, while A6 includes the dotted input. Overall decryption of A5, A6, and N2 will recompute and verify the tag first, only proceeding with further decryption of C if this verification is successful.

A keyed function $\mathcal{F}$ is called pseudorandom if for a random drawn secret key, it behaves as if it were a random function. This is formalized in Def. 4 below. It is easy to build a MAC from a pseudorandom function $\mathcal{F}$, simply by setting $\mathcal{T}_k = \mathcal{F}_k$ and $\mathcal{V}_k(M,T) = \top \iff \mathcal{F}_k(M) = T$. In that case a distinguishing adversary against the MAC in the sEUF–CMA game can be turned in an equally successful adversary against the PRF property of $\mathcal{F}$ (without incurring any significant overhead in the reduction).

Definition 4 (Pseudo Random Functions). *Let $\mathcal{F} : \mathsf{K} \times \mathsf{M} \rightarrow \mathsf{T}$ be a function, and $\mathbb{A}$ an adversary who does not forward queries between their oracles. Then, the* PRF *advantage of $\mathbb{A}$ against $\mathcal{F}$ is*

$$\mathbf{Adv}_{\mathcal{F}}^{\mathrm{PRF}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathcal{F}_k, \mathcal{F}_k \\ \$, \mathcal{F}_k\end{pmatrix} .$$

We say $\mathcal{F}$ is a PRF *if the PRF advantage is small for all adversaries running within reasonable resources.*

Generic composition for nAE. NRS [38] investigated how to construct an nAE scheme by composing two PRFs with an ivE scheme. The IV of the ivE scheme is derived from the nAE's inputs using the first PRF call; the optional second PRF call may be used to create an authentication tag. Different schemes emerge by changing which variables are provided to each of the components. NRS identify eight schemes, dubbed A1–A8, with strong security bounds. For a further four schemes (A9–A12) neither strong security bounds nor insecurity was established. Additionally, NRS investigated mechanisms for combining a PRF with an nE scheme. Three schemes (N1–N3) were found secure, with that of a fourth (N4) remaining unresolved.

Figure 2's middle panel shows the schemes A5 and A6. For these two schemes, as well as for N2 (on the left), the ciphertext is input to the second PRF, which means they classify as Encrypt-then-MAC (EtM). The schemes A4, A7–A12, as well as N3 and N4 only use a single PRF and release the IV as tag; for that reason we refer to them as MAC-then-Encrypt (MtE). Finally, the schemes A1–A3 and N1 use two PRFs that can be called in parallel, leading to their classification as Encrypt-and-MAC (E&M), (We refer to NRS for full descriptions and graphical illustrations of all schemes mentioned above).

3 Security Notions in the Presence of Leakage

Authenticated encryption, as defined above, is deterministic. In a leakage-free setting, this is provides a stronger notion than the older probabilistic notion of encryption (as implicitly still used for ivE). When introducing leakage, deterministic schemes are problematic both from a practical and a theoretical perspective.

On the one hand, a practical side-channel attack such as differential power analysis can effectively recover keys from unprotected blockciphers and their AE modes with near certainty. Randomized masking based on secret sharing is one of the main countermeasures against these attacks.

On the other hand, theoretical leakage is often modelled as a function on the inputs of the computation, which will include the key. If with each invocation of the scheme an adversary can let the scheme leak a different key bit of its choice, the full key is easily recovered. To prevent such devastating yet simple leakage, a typical design strategy is to split the key in two shares and update the shares on-the-fly using fresh randomness, mimicking the practical approach.

One mechanism that can avoid relying on randomness is to instead use a leak-free component [57]. Our notation does support this (by suitable choice of leakage set), but such components are between hard and impossible to instantiate in practice [33], so we do not wish to be restricted to this case. We leave open potential solutions relying on synchronized states between encryption and decryption as such synchronization can be difficult to maintain, thereby restricting applicability of the model. However, in the specific context of secure channels, synchronization might not be too onerous.

3.1 Implementations versus Functions

In both the practical and the theoretical approach mentioned above a deterministic scheme is implemented in a randomized fashion in order to provide resistance against leakage. Therefore, when arguing about leakage, we will need to make a distinction between the scheme (a deterministic function) and its implementation.

For our definition of the implementations of a function we take our cue from the secret-sharing approach, where a redundant representation of the key is used and this representation is rerandomized as part of the implementation. To enable this rerandomization, we provide the implementation of a function with explicit randomness in Definition 5 below.

Definition 5 (Implementation of f). *An implementation of a function $f : \mathsf{K} \times X \to Y$ is a deterministic function $\boldsymbol{f} : \mathbf{K} \times X \times \mathsf{R} \to \mathbf{K} \times Y$ along with a probabilistic key initialisation function $\iota : \mathsf{K} \to \mathbf{K}$ such that $\iota(k) = \iota(l) \iff k = l$. We define the inverse of ι as the function $\iota^{-1} : \mathbf{K} \to \mathsf{K}$ such that $\iota^{-1}(\boldsymbol{k}) = k$ if there exists k such that $\iota(k) = \boldsymbol{k}$, and $\perp$ otherwise.*

The implementation is correct *iff for all $k \in \mathsf{K}, x \in X$, and $r \in \mathsf{R}$, setting $\boldsymbol{k} \leftarrow \iota(k)$ and $(\boldsymbol{k}', y) \leftarrow \boldsymbol{f}(\boldsymbol{k}, x; r)$ guarantees both $y = f(k, x)$ and $\iota^{-1}(\boldsymbol{k}') = k$.*

To guide the reader, we use a bold font to denote either the implementation of a function or the representation of a key used by the implementation. The initial representation of the key is generated using the function ι, which maps a key $k \in \mathsf{K}$ to a suitable representation $\boldsymbol{k} \in \mathbf{K}$ for the implementation. We assume that ι is performed only once, and in a leak-free manner, during setup (straight after key generation). Moreover, its inverse ι^{-1} induces an equivalence relation on the space $\mathbf{K}$; in other words, the implementation keys $\boldsymbol{k}$ can be thought of as alternative representations of the key.

Discussion. Correctness implies that an implementation is identical to the original function when restricted to the second output and that the new key representation $\boldsymbol{k}'$ is equivalent to the initial one $\boldsymbol{k}$. We make no demands of $\boldsymbol{k}$ or $\boldsymbol{k}'$ beyond these, so it is permissible to set $\boldsymbol{k} = \boldsymbol{k}' = k$ and thus recover the traditional syntax. Our security definitions will be such that for correct schemes and assuming 'trivial' leakage, the corresponding leak-free security notions from the preceding section will emerge.

Definition 5 can be linked to practice in a straightforward manner. Recall that practical implementations of blockciphers often use masking based on secret sharing schemes. In this case, the implementation of the blockcipher describes how to evaluate the blockcipher based on the shares of the key as well as how the sharing is refreshed using external randomness r. Furthermore, ι is exactly the function that creates the initial secret sharing of the key.

Syntactically the implementation $\boldsymbol{f}$ may appear stateful: after all they take in some $\boldsymbol{k}$ and output an updated $\boldsymbol{k}'$ for the next invocation. However, since the implementation is of a stateless function f, there is no need to synchronize state between communication parties. Instead, each party can use its own, independent representation of the key.

Implementation of an nAE Scheme. For concreteness, we now explicitly define the implementation of an nAE scheme, where we assume that Enc and Dec use the same representations $\mathbf{K}$ and mapping ι for the key, corresponding to them both being functionalities derived from the same device.

By correctness of the implementation, one can see that the ciphertext output by **Enc** (resp. message by **Dec**) will always be independent of the randomness, since they are equal to the corresponding output of Enc (resp. Dec). Definitions for the implementations of other security primitives are written accordingly.

Definition 6 (AE Implementation). *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be an authenticated encryption scheme. An* AE implementation *is a pair of deterministic functions*

$$\begin{aligned}\mathbf{Enc}&\colon \mathbf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \times \mathsf{R} \to \mathbf{K} \times \mathsf{C}\\ \mathbf{Dec}&\colon \mathbf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \times \mathsf{R} \to \mathbf{K} \times (\mathsf{M} \cup \{\perp\})\end{aligned}$$

along with $\iota\colon \mathsf{K} \times \mathsf{R} \to \mathbf{K}$ *such that* $\iota(k, r) = \iota(l, s)$ *iff* $k = l$ *and* $\iota^{-1}\colon \mathbf{K} \to \mathsf{K}$ *such that* $\iota^{-1}(()\boldsymbol{k}) = k$ *if there exists* $r \in \mathsf{R}$ *such that* $\iota(k, r) = \boldsymbol{k}$*. The implementation is* correct *iff for any* k, N, A, M, C, r *from the appropriate spaces and* $\boldsymbol{k} \leftarrow\!\!\$\ \iota(k)$*, setting*

$$(\boldsymbol{k}', C') \leftarrow \mathbf{Enc}(\boldsymbol{k}, N, A, M; r) \text{ and } (\boldsymbol{k}'', M') \leftarrow \mathbf{Dec}(\boldsymbol{k}, N, A, C; r),$$

the following properties hold:

$$\begin{aligned}&k = \iota^{-1}(\boldsymbol{k}) = \iota^{-1}(\boldsymbol{k}') = \iota^{-1}(\boldsymbol{k}'')\\ &C' = \mathsf{Enc}(k, N, A, M) \text{ and } M' = \mathsf{Dec}(k, N, A, C)\ .\end{aligned}$$

3.2 What Constitutes Leakage

The input syntax of a leakage function matches to the input syntax of the implementation $\boldsymbol{f}$ that it relates to. A leakage set is a collection of leakage functions for an implementation.

Definition 7 (Leakage Functions). *A* leakage function *of an implementation* $\boldsymbol{f}\colon \mathbf{K} \times X \times \mathsf{R} \to \mathbf{K} \times Y$ *is a function* $L\colon \mathbf{K} \times X \times \mathsf{R} \to \mathsf{L}$ *for some output leakage space* L*. A* leakage set *of an implementation* $\boldsymbol{f}$ *is a set of leakage functions.*

The choice of leakage set should contain all plausible (functions of) inputs to the implementation that an adversary can compute, and may be probabilistic. This might include functions of any intermediate variables, since these are computable from the inputs simply by simulating the construction. Broadly speaking, the larger the leakage set the more powerful the adversary is likely to be. The leakage set $\emptyset$ allows us to model the leak-free case. Technically we define it to be the set containing just the null function, meaning the adversary can always select a leakage function, thus maintaining the correct syntax for our security games.

3.3 Security Notions including Leakage

We are now in a position to define the security of an implementation in the presence of leakage. We do so by reframing the classical notions given to work on the implementation of a function, and by extending the notions such that the honest oracles are allowed to leak. The adversary wins the game if they can distinguish whether their leak-free challenge oracles implement the scheme honestly or are idealised. We differentiate our notions from the classic variant by prefixing an "L", for *leakage*.

In the classical setting, each oracle simply evaluates the appropriate function with the games secret key. For an implementation, a similar, but slightly more complicated, approach is required. The oracle

```
function $\ell[\boldsymbol{\mathcal{E}}]_k(M; L)$
    $r \leftarrow\$ \mathsf{R}$
    $\Lambda \leftarrow L(\boldsymbol{k}, M; r)$
    $C, \boldsymbol{k} \leftarrow \boldsymbol{\mathcal{E}}(\boldsymbol{k}, M; r)$
    return $(C, \Lambda)$

function $\ell[\boldsymbol{\mathcal{D}}]_k(C; L)$
    $r \leftarrow\$ \mathsf{R}$
    $\Lambda \leftarrow L(\boldsymbol{k}, C; r)$
    $M, \boldsymbol{k} \leftarrow \boldsymbol{\mathcal{D}}(\boldsymbol{k}, C; r)$
    return $(\perp, \Lambda)$

function $\ell[\mathbf{Enc}]_k(N, A, M; L)$
    $r \leftarrow\$ \mathsf{R}$
    $\Lambda \leftarrow L(\boldsymbol{k}, N, A, M; r)$
    $C, \boldsymbol{k} \leftarrow \mathbf{Enc}(\boldsymbol{k}, N, A, M; r)$
    return $(C, \Lambda)$

function $\ell[\mathbf{Dec}]_k(N, A, C; L)$
    $r \leftarrow\$ \mathsf{R}$
    $\Lambda \leftarrow L(\boldsymbol{k}, N, A, C; r)$
    $M, \boldsymbol{k} \leftarrow \mathbf{Dec}(\boldsymbol{k}, N, A, C; r)$
    return $(M, \Lambda)$
```

Fig. 3: Honest leakage oracles an adversary may use to help them distinguish. All inputs are taken from the appropriate spaces, with leakage functions chosen from the relevant leakage set. For $\mathcal{L}_{\mathsf{E}}$-IND–CPLA, the adversary has access to $\ell[\boldsymbol{\mathcal{E}}]_k$, and for the augmented notion $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-IND-aCPLA they are also given very limited access to $\ell[\boldsymbol{\mathcal{D}}]_k$. LAE security, $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$-LAE provides access to $(\ell[\mathbf{Enc}]_k, \ell[\mathbf{Dec}]_k)$.

```
function $\ell[\boldsymbol{\mathcal{T}}]_k(M; L)$
    $r \leftarrow\$ \mathsf{R}$
    $\Lambda \leftarrow L(\boldsymbol{k}, M; r)$
    $T, \boldsymbol{k} \leftarrow \boldsymbol{\mathcal{T}}(\boldsymbol{k}, M; r)$
    return $(T, \Lambda)$

function $\ell[\boldsymbol{\mathcal{V}}]_k(M, T; L)$
    $r \leftarrow\$ \mathsf{R}$
    $\Lambda \leftarrow L(\boldsymbol{k}, M, T; r)$
    $V, \boldsymbol{k} \leftarrow \boldsymbol{\mathcal{V}}(\boldsymbol{k}, M, T; r)$
    return $(V, \Lambda)$
```

Fig. 4: Honest leakage oracles an adversary may use to help them distinguish. All inputs are taken from the appropriate spaces, with leakage functions chosen from the relevant leakage set. $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$-LMAC security gives access to $(\ell[\boldsymbol{\mathcal{T}}]_k, \ell[\boldsymbol{\mathcal{V}}]_k)$. Since PRFs and the tagging function of a MAC have the same syntax, the LPRF game provides access to $\ell[\boldsymbol{\mathcal{F}}]_k$, which is identical to $\ell[\boldsymbol{\mathcal{T}}]_k$.

must draw randomness, and provide this to the implementation to update the key representation. This same randomness, along with all other inputs, must be provided to the leakage function. The new key must then be stored, and the two outputs returned to the adversary. For any implementation $\boldsymbol{f}$, the corresponding leakage oracle is denoted $\ell[\boldsymbol{f}]_k$, when initialised with key k. A code-based description of this is given in Figure 3, along with examples for some standard primitives.

As in the leakage free definitions, security is taken over the randomness of the initial keys, and of the oracles. Notice that this choice includes the sampling from R. We assume the adversary only ever makes queries for which his inputs are selected from the appropriate spaces. For leakage, this means some leakage set that will be specified in the security notion.

Definition 8 (AE security against leakage). *Let* $(\mathbf{Enc}, \mathbf{Dec})$ *be an implementation of an authenticated encryption scheme* $\mathsf{Enc}, \mathsf{Dec}$*, and* $\mathbb{A}$ *an adversary who does not forward queries to or from his first or second oracles (and thus does not repeat such queries). Then, the* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$–LAE advantage *of an adversary* $\mathbb{A}$ *against* $(\mathbf{Enc}, \mathbf{Dec})$ *under leakage* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$ *is*

$$\mathbf{Adv}^{\mathrm{LAE}}_{\mathbf{Enc},\mathbf{Dec};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathsf{Enc}_k, \mathsf{Dec}_k, \ell[\mathbf{Enc}]_k, \ell[\mathbf{Dec}]_k \\ \$ \quad , \quad \perp \quad , \ell[\mathbf{Enc}]_k, \ell[\mathbf{Dec}]_k\end{pmatrix}.$$

The implementation is $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$-nLAE secure*, or simply* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$-nLAE*, if this advantage is small for all reasonably resourced, nonce-respecting adversaries, with* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$-mrLAE *defined similarly.*

Definition 9 (Encryption security against leakage). *Let* $\boldsymbol{\mathcal{E}}$ *be an implementation of an encryption scheme* $\mathcal{E}$*, and* $\mathbb{A}$ *an adversary who never forwards queries to or from his first oracle (and thus does not repeat first oracle queries). The* $\mathcal{L}_{\mathsf{E}}$*-IND–CPLA advantage (named for chosen-plaintext-with-leakage-attack) of* $\mathbb{A}$ *against* $\boldsymbol{\mathcal{E}}$ *is*

$$\mathbf{Adv}^{\mathrm{IND-CPLA}}_{\boldsymbol{\mathcal{E}};\mathcal{L}_{\mathsf{E}}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathcal{E}_k, \ell[\boldsymbol{\mathcal{E}}]_k \\ \$ \ , \ell[\boldsymbol{\mathcal{E}}]_k\end{pmatrix}.$$

*We say $\boldsymbol{\mathcal{E}}$ is an $\mathcal{L}_{\mathsf{E}}$-*nLE *implementation if the $\mathcal{L}_{\mathsf{E}}$-IND–CPLA advantage is small for all nonce-respecting adversaries running within reasonable resources, with $\mathcal{L}_{\mathsf{E}}$-*ivLE *defined similarly.*

We next provide an additional encryption notion, IND–aCPLA, that will be required for our composition results later. It describes a modified version of the IND–CPLA game in which the adversary is also allowed leakage from the decryption implementation, but *only* on ciphertexts they have previously received from $\ell[\boldsymbol{\mathcal{E}}]_k$. At first glance, this appears to be more similar to an IND–CCA style notion, but we emphasise this is not the case since the possible decryption queries are heavily restricted. Thus it should thought of as IND–CPA under the most general form of leakage. Indeed, when the leakage sets are set to be empty, the resulting security notion is equivalent to IND–CPA.

Definition 10 (Augmented CPLA security). *Let $(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}})$ be an implementation of an encryption scheme, $\mathbb{A}$ an adversary who does not forward queries to or from his first oracle, and only makes queries to their third oracle that were forwarded from the second. Then the $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-IND–aCPLA advantage of $\mathbb{A}$ against $\boldsymbol{\mathcal{E}}$ is*

$$\mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{E}},\mathcal{L}_{\mathsf{D}}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta} \begin{pmatrix} \mathcal{E}_k, \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k \\ \$ \,, \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k \end{pmatrix} .$$

*The implementation is $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-*nLE *if the $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-IND–aCPLA advantage is small for all nonce-respecting adversaries running within reasonable resources, $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-*ivLE *defined similarly.*

The IND–aCPLA notion is required for the general composition. The goal is to construct an LAE scheme from an ivLE scheme (and other components). However, for decryption of the LAE scheme to leak (as we want the leakage to be as powerful as possible), the decryption of ivLE scheme would have to leak. The IND–CPLA security notion does not capture this. Consider an IND–CPA scheme where encryption does not leak, but the leakage from decrypting the zero string returns the key. Clearly the scheme is also IND–CPLA but will trivially break when the adversary is given decryption leakage. The IND–aCPLA notion is trying to capture that decryption "does not leak too much information", so that limited decryption queries made by the LAE scheme will be able to leak.

Against many natural choices of leakage sets, $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-IND–aCPLA and $\mathcal{L}_{\mathsf{E}}$-IND–CPLA are equivalent, since the encryption oracle often suffices to simulate any leakage from decryption. In the nonce-abusing setting, where the adversary is free to select nonces however they wish, there exists an explicit reduction showing this.

In the nonce respecting or iv respecting scenarios such a general reduction is not possible, because there is no way to allow the adversary to use the same nonce multiple times, something a decryption oracle would allow. If the leakage is independent of the nonce (for example) similar results can be recovered, but these are much more restrictive scenarios. It is an interesting open problem to describe sets $\mathcal{L}_{\mathsf{ED}}$ that are in some sense "minimal" for various pairs of leakage sets $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$ taken from some general function classes.

LMAC and LPRF. Here we give the PRF and MAC notions a similar treatment to the encryption definitions given previously, enhancing the standard definitions by allowing leakage off the honest oracles.

The strong Existential unforgibility under chosen message with leakage attack notion for LMAC security strengthens both the classical definition, and the definition Martin *et al.* [34], who only allow tagging to leak; setting $\mathcal{L}_{\mathsf{V}} = \emptyset$ recovers their definition. They mention this more general definition but do not explicitly define it. We cast it as a distinguishing game, rather than the more traditional computational game ("adversary must forge a tag"), but it is straightforward to see this is equivalent (even in the face of leakage).

The LPRF definition strengthens the definition of Dodis and Pietrzak [16], and Faust *et al.* [18] in that both the leakage functions and the inputs can be chosen adaptively based on outputs already seen by the adversary.

Definition 11 (MAC security against leakage). *Let $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ be an implementation of a MAC $(\mathcal{T}, \mathcal{V})$, and $\mathbb{A}$ an adversary who does not forward queries from his second oracle to the first. Then the $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$-sEUF-CMLA advantage of $\mathbb{A}$ against $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ under leakage $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$ is*

$$\mathbf{Adv}^{\mathrm{sEUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_{\mathsf{T}},\mathcal{L}_{\mathsf{V}}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta} \begin{pmatrix} \mathcal{V}_k, \ell[\boldsymbol{\mathcal{T}}]_k, \ell[\boldsymbol{\mathcal{V}}]_k \\ \bot, \ell[\boldsymbol{\mathcal{T}}]_k, \ell[\boldsymbol{\mathcal{V}}]_k \end{pmatrix}.$$

We say $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ is a secure $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$-LMAC if this advantage is small for all adversaries running within reasonable resources.

Definition 12 (PRF security against leakage). *Let $\boldsymbol{\mathcal{F}}$ be an implementation of a function $\mathcal{F}$, and $\mathbb{A}$ an adversary who never forwards or repeats queries. Then the $\mathcal{L}_{\mathsf{F}}$-PRLF advantage of $\mathbb{A}$ against $\boldsymbol{\mathcal{F}}$ under leakage $\mathcal{L}_{\mathsf{F}}$ is*

$$\mathbf{Adv}^{\mathrm{PRLF}}_{\boldsymbol{\mathcal{F}};\mathcal{L}_{\mathsf{F}}}(\mathbb{A}) := \underset{\mathbb{A}}{\Delta} \begin{pmatrix} \mathcal{F}_k, \ell[\boldsymbol{\mathcal{F}}]_k \\ \$, \ell[\boldsymbol{\mathcal{F}}]_k \end{pmatrix}.$$

We say $\boldsymbol{\mathcal{F}}$ is a secure $\mathcal{L}_{\mathsf{F}}$-LPRF if this advantage is small for all adversaries running within reasonable resources.

4 Applying LAE to attacks in theory and practice

A security framework is not much use if it does not highlight the difference between schemes for which strong security results are known, and those against which efficient attacks exist. In this section we discuss the types of leakage normally considered within the literature. We show how previous leakage models can be captured by our leakage set style notion. In the literature there is focus on two types of leakage; protocol leakage (by the AE literature) and side channel leakage (by the leakage resilient literature). We believe that these two notions are highly related and thus we discuss how to capture both. For example, termination of an algorithm at different points (distinguishable decryption failures) is normally detected by a side-channel; timing can be used to capture this if the failures terminate the algorithm at different points in time and power can be used to detect if conditional branches were taken.

After first describing generic methods to recast existing leakage resilience work within our general framework, we consider a few concrete examples of this, describing existing attacks within our setting.

4.1 Theoretical leakage models

We observe that our model is in many ways the most general possible, and that many previous leakage notions can be captured as version of the $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-LAE security game for suitable choice of leakage sets $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$. Reassuringly, by setting $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}}) = (\emptyset, \emptyset)$ we recover the traditional leakage-free security notions, with $(\emptyset, \emptyset)$-nLAE equivalent to nAE, and both $\emptyset$-IND–CPLA and $(\emptyset, \emptyset)$-IND–aCPLA equivalent to IND–CPA, meaning a secure nE scheme is $\emptyset$-nLE secure.

The deterministic decryption leakage notions from the AE literature can be recovered by choosing the appropriate leakage set. The SAE framework generalises both the RUP model, and the (nonce-based analogues of the) Distinguishable Decryption Failure notions of Boldyreva *et al.* [2,4,12]. The security notions are parametrised by a deterministic decryption leakage function Λ, corresponding to security under the leakage sets $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}}) = (\emptyset, \{\Lambda\})$. Thus the strongest notions available in these settings are equivalent to $(\emptyset, \{\Lambda\})$–LAE. Several of their weaker notions translate to the corresponding weakening of this, including authenticity under deterministic leakage, (known variously as CTI–sCPA, INT–RUP or an extended form of INT–CTXT), which translates to a variant of $(\emptyset, \{\Lambda\})$–LAE in which the adversary cannot query the encryption challenge oracle (and thus does not interact with either $\mathcal{E}_k$ or \$).

In the simulatable leakage model (*e.g.* [54]), the adversary receives leakage in addition to their query, but is restricted to leakage functions that can be simulated without the key. The simulatable model considered by Standaert *et al.* (for example) can be captured by our model by having set of leakage functions contain the single function which provides the power trace to the adversary. The auxiliary

```
**function** Enc$(N, A, M)$
    $A_1||\dots||A_a \leftarrow \text{Parse}(A)$
    $M_1||\dots||M_m \leftarrow \text{Parse}(M)$
    $\text{ctr} \leftarrow N||0^{31}||1$
    $H \leftarrow \text{E}_k(0^{128})$
    $T_0 \leftarrow 0^{128}$
    **for** $i = 1, \dots, a$ **do**
        $T_i \leftarrow (T_{i-1} + A_i) \bullet H$
    **for** $i = 1, \dots, m$ **do**
        $C_i \leftarrow M_i \oplus \text{E}_k(\text{ctr} + i)$
        $T_{a+i} \leftarrow (T_{a+i-1} + C_i) \bullet H$
    $T_{a+m+1} \leftarrow (T_{a+m} + (a||m)) \bullet H$
    $T \leftarrow T_{a+m+1} \oplus \text{E}_k(\text{ctr})$
    $C \leftarrow C_1||\dots||C_m||T$
    **return** $C$
```

```
**function** $\ell[\textbf{Enc}]_k(N, A, M; L)$
    $r \leftarrow_\$ \mathsf{R}$
    $A_1||\dots||A_a \leftarrow \text{Parse}(A)$
    $M_1||\dots||M_m \leftarrow \text{Parse}(M)$
    $\text{ctr} \leftarrow N||0^{31}||1$
    $H \leftarrow \text{E}_k(0^{128})$
    $T_0 \leftarrow 0^{128}$
    **for** $i = 1, \dots, a$ **do**
        $T_i \leftarrow (T_{i-1} + A_i) \bullet H$
    **for** $i = 1, \dots, m$ **do**
        $C_i \leftarrow M_i \oplus \text{E}_k(\text{ctr} + i)$
        $T_{a+i} \leftarrow (T_{a+i-1} + C_i) \bullet H$
    $T_{a+m+1} \leftarrow (T_{a+m} + (a||m)) \bullet H$
    $T \leftarrow T_{a+m+1} \oplus \text{E}_k(\text{ctr})$
    $C \leftarrow C_1||\dots||C_m||T$
    $\Lambda \leftarrow L(k, N, A, M; r)$
    **return** $(C, \Lambda)$
```

```
**function** $\ell[\textbf{Dec}]_k(N, A, M; L)$
    $r \leftarrow_\$ \mathsf{R}$
    $C'||T \leftarrow C$ with $|T| = 128$
    $A_1||\dots||A_a \leftarrow \text{Parse}(A)$
    $C_1||\dots||C_m \leftarrow \text{Parse}(C)$
    $\text{ctr} \leftarrow N||0^{31}||1$
    $H \leftarrow \text{E}_k(0^{128})$
    $T_0 \leftarrow 0^{128}$
    **for** $i = 1, \dots, a$ **do**
        $T_i \leftarrow (T_{i-1} + A_i) \bullet H$
    **for** $i = 1, \dots, m$ **do**
        $M_i \leftarrow C_i \oplus \text{E}_k(\text{ctr} + i)$
        $T_{a+i} \leftarrow (T_{a+i-1} + C_i) \bullet H$
    $T_{a+m+1} \leftarrow (T_{a+m} + (a||m)) \bullet H$
    **if** $T \neq T_{a+m+1} \oplus \text{E}_k(\text{ctr})$ **then**
        $M \leftarrow \perp$
    **else**
        $M \leftarrow M_1||\dots||M_m$
    $\Lambda \leftarrow L(k, N, A, C; r)$
    **return** $(M, \Lambda)$
```

Fig. 5: A standard implementation of GCM, with $\mathsf{N} = \{0,1\}^{96}$, $\mathsf{K} = \{0,1\}^{128}$ and $\mathsf{M} = \mathsf{C} = \mathsf{A} = \{0,1\}^*$, providing the plaintext/ciphertext plus authenticated data is at most 2^{32} blocks. The first column defines the scheme, while the second and third give leakage oracles instantiating it. The function "Parse" splits a string into blocks of 128 bits, with the final block possibly incomplete; E_k is AES–128 with key k. The xor different length strings returns a string of the shorter length, while $\bullet$ and $+$ denote the binary operations of GF(2^{128}).

input model [15] gives the adversary the output of a hard to invert function applied to the key, alongside the normal security notion interactions. The only computation leaks model [37] (discussed in more detail in Section 5.1) restricts the adversary to leakage functions that can be locally computed: any step of the algorithm can only leak on variables being used at that point. In the following sections we show how this leakage set can be defined for our given constructions.

In the probing model [24] the adversary can gain access to the values of t of the internal wires from the computation. A scheme is secure if an adversary with the knowledge of t internal wires can do no better than if they had access to the function in a black box manner. If there are n internal wires, this leakage can be captured by our set notation by constructing a set with n choose t leakage functions, each giving the complete value of the relevant wires.

Our leakage sets incorporate the bounded leakage model (*e.g.* [22, 26, 30]) by restricting the set of allowable adversaries to those who only make sufficiently few queries to the leakage oracles.

4.2 A practical example from the literature

As a concrete example of how to set an existing attack within the framework, consider the attack against the well-known AES–GCM [36] AEAD mode by Belaïd, Fougue and Gérard [5]. The authors observe that that the least significant bit (lsb) of a vector's Hamming weight is simply a linear combination of its bits. Therefore, if the Hamming weight (or an approximation of it) is available, then (some approximation) of this linear combination can be deduced. Based on this, they construct an attack on the authentication key, using leakage from the first polynomial multiplication.

The GCM encryption routine and the relevant leakage oracle are given in Figure 5. Note that as the implementation under attack was not randomised, the oracle's randomness r is not used. To apply the attack, we require that $L(k, N, A, M; r) = \text{HW}(A_1 \bullet \text{E}_k(0^{128})) \oplus \varepsilon$ is an element of the encryption leakage set $\mathcal{L}_\mathsf{E}$, or that the corresponding function $L(k, N, A, C; r)$ is in $\mathcal{L}_\mathsf{D}$. Within GCM, the value $H = \text{E}_k(0^{128})$ is the authentication key, so this leakage function calculates its Hamming weight (HW) after multiplication by a known quantity A_1, before adding some Gaussian noise ε to better describe experimental error.

The adversary makes a series of queries to their honest oracles to acquire this leakage, and then collects the least significant bit of the each leakage query, along with the corresponding value A_1, to form a system of (noisy) linear equations. In the most naive case, if it is possible to guess where the noise causes errors in the linear equations, they can be solved in the standard manner to find $H = \mathrm{E}_k(0^{128})$, which completely breaks the security of the scheme. For 6 errors this requires approximately 2^{32} effort. Other elements of the original work expand upon this, providing more involved techniques for reducing the amount of work required to solve this noisy system.

Overall then, this attack demonstrates that the implementation of AES-GCM described in Figure 5 is neither $(\emptyset, \{L\})$–nLAE nor $(\{L\}, \emptyset)$–nLAE.

5 Generic Composition for LAE

5.1 Modelling Composed Leakage

Our challenge is to establish to what extent the NRS schemes remain secure when taking leakage into account.[3] Ideally, we would like to claim that if both the ivE and the PRFs are secure in the presence of leakage, then so will the composed nAE be. To make such a statement precise, the leakage classes involved need to be specified. We opt for an approach where the leakage classes for the components are given (and can be arbitrary) and then derive a leakage class for the resulting nAE for which we can prove security.

Encryption leakage. In a nutshell, we define the leakage of the composition as the composition of the leakage. As an example, consider A5 (Figure 2). When encrypting, the leakage may come from any of the components: the PRF $\boldsymbol{\mathcal{F}}$ may leak some information $L_F(\boldsymbol{k}_F, N; r_F)$; the IV-encryption routine $\mathbf{iv}\boldsymbol{\mathcal{E}}$ might leak some information $L_E(\boldsymbol{k}_E, I, M; r_E)$; the Tag function $\boldsymbol{\mathcal{T}}$ may leak some information $L_T(\boldsymbol{k}_M, N, A, C_e; r_M)$. To ease notation, we will use the shorthand $L_F(\star)$, $L_E(\star)$, and $L_T(\star)$ respectively for these leakages. In that case, we say that the leakage on the authenticated encryption operation as a whole consists of the triple $(L_F(\star), L_E(\star), L_T(\star))$. Under the hood, this implies some parsing and forwarding of the AE's key $(\boldsymbol{k}_F, \boldsymbol{k}_E, \boldsymbol{k}_M)$, randomness (r_F, r_E, r_M) and inputs N, A, M, including the calculated values I and C_e, to the component leakage functions L_F, L_E, and L_T.

Expanding the above to classes of functions is as follows. Let $\mathcal{L}_{\mathsf{F}}, \mathcal{L}_{\mathsf{E}}$, and $\mathcal{L}_{\mathsf{T}}$ be the respective leakage classes for $\boldsymbol{\mathcal{F}}, \mathbf{iv}\boldsymbol{\mathcal{E}}$, and $\boldsymbol{\mathcal{T}}$. Then the leakage class $\mathcal{L}_{\mathsf{Enc}}$ for the resulting authenticated encryption scheme is defined as $\{(L_F, L_E, L_T) | L_F \in \mathcal{L}_{\mathsf{F}}, L_E \in \mathcal{L}_{\mathsf{E}}, L_T \in \mathcal{L}_{\mathsf{T}}\}$. Since an adversary has to select a leakage function in $\mathcal{L}_{\mathsf{Enc}}$ the moment it queries the encryption oracle, it will not be possible to adaptively select for instance the leakage function L_T based on the leakage received from L_E *of that encryption query*.

Decryption leakage. In order to describe leakage from decryption, we take a closer look at the role of the two PRFs in the generic constructions. The first one, $\boldsymbol{\mathcal{F}}$, computes the initial vector which is needed both for encryption and decryption. This makes it inevitable that during decryption it is again computed as a PRF, presumably using the same implementation $\boldsymbol{\mathcal{F}}$. On the other hand, the second PRF, $\boldsymbol{\mathcal{T}}$, is used to create a tag T during encryption. Normally during decryption one would recompute the tag (again using $\boldsymbol{\mathcal{T}}$) and check whether the recomputed tag T' equals the received tag T. Yet, in the leakage setting this approach is problematic: T' is the correct tag and its recomputation might well leak it, even when used (repeatedly) to check an incorrect and completely unrelated T. Hence, during decryption we will not use a recompute-and-check model, but rather refer directly to a tag-verification implementation $\boldsymbol{\mathcal{V}}$ (that hopefully leaks less).

When considering the decryption leakage of A5, we will assume that, on invalid ciphertexts, the computation terminates as soon as the verification algorithm returns $\perp$. This implies that for invalid

[3] Obviously, leakage is not suddenly going to promote any of the insecure schemes into a secure one.

Structure	Leakage	Inverse	Inverse Leakage
MtE	$L_T(\star)$, $L_F(\star)$, $L_E(\star)$	DtV	$L_F(\star)$, $L_D(\star)$, $L_V(\star)$
M&E	$L_T(\star)$, $L_F(\star)$, $L_E(\star)$		
EtM	$L_F(\star)$, $L_E(\star)$, $L_T(\star)$	D&V	$L_F(\star)$, $L_D(\star)$, $L_V(\star)$
		VtD	$\begin{cases} L_V(\star) & \text{if } \mathcal{V}(\star) = \bot \\ L_V(\star), L_F(\star), L_D(\star) & \text{if } \mathcal{V}(\star) = \top \end{cases}$

Fig. 6: The structure of a leakage function from a composition scheme based on the order of its primitives. The exact input parameters to the leakage function vary per scheme, so have been replaced with $\star$: the different $\star$ variables are not the same. On the left are the encryption structures MtE, M&E and EtM, along with the associated leakage function. The right gives the associated inverse: DtV (Decrypt then Verify) is the only way of inverting MtE or M&E schemes. EtM schemes can be inverted in any order, as DtV, D&V (Decrypt and Verify) or VtD (Verify then Decrypt). All constructions have the same encryption leakage, and most have the same decryption leakage. The only one that is different is an EtM–VtD scheme, where the decryption leakage format depends on the validity of the ciphertext.

ciphertexts only leakage on $\boldsymbol{\mathcal{V}}$ will be available, whereas for valid ciphertexts all three components ($\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{F}}$, and $\mathbf{iv}\boldsymbol{\mathcal{E}}$) might leak.

Overview and interpretation. Recall that we divided the NRS schemes in three categories: MtE, M&E, and EtM. Figure 6 shows how the composed leakage will leak for each of these schemes. For completeness, we also listed the leakage for the EtM scheme (such as A5) in case full decryption will always take place, even for invalid ciphertexts (where one could have aborted early).

Our choice to model the leakage from the authenticated encryption scheme as completely separate components from the three underlying primitives is rooted in the assumption that only computation leaks. This assumption was first formalized by Micali and Reyzin [37] and, although there are counterexamples to the assumption at for instance the gate level [45], we believe that implementations of the three primitives result in large enough physical components, which can be suitably segragated to avoid cross-leakage.

Leakage on the wire (for instance of the initial vector I) can be captured as leakage of the PRF computing the I or alternatively as that of the ivE. In particular, by letting the decryption of the $\mathbf{iv}\boldsymbol{\mathcal{E}}$ component leak its full output (while not allowing any further leakage), we capture the release of unverified plaintext. Furthermore, distinguishable decryption failures on MtE and M&E schemes invariably arise from verification, which might incorporate a padding check as well. This is modelled by allowing $\mathcal{V}$ to leak, but not any of the other components.

5.2 MAC-and/then-Encrypt are Brittle under Leakage

For schemes where the plaintext is input to the MAC (i.e. MtE and M&E schemes), decryption is inevitably of the form DtV. Consequently, during decryption a purported message M is computed before the tag can be verified. Leaking this message M corresponds to the release of unverified plaintext [2], but even more modest leakage, such as the first bit of the candidate message, can be insecure as we show by the following example.

Let us assume for a moment that the encryption routine $\mathbf{iv}\boldsymbol{\mathcal{E}}$ is online, so that reencrypting a slightly modified plaintext using the same I will only affect a change in the ciphertext after the modification in the plaintext. CBC and CFB modes are well-known examples of online $\mathbf{iv}\boldsymbol{\mathcal{E}}$ schemes. Additionally, assume that $\mathbf{iv}\boldsymbol{\mathcal{E}}$'s decryption routine indeed leaks the first bit of the message. Then the authenticated encryption scheme is not secure in the presence of leakage (for the leakage class derived according to the principles from the previous subsection), which an attack demonstrates.

The adversary first submits a message M to its challenge encryption oracle, receiving either a ciphertext C which either is an encryption $\mathcal{E}_k(M||T)$ or (in the ideal world) a uniformly random string. The adversary subsequently queries its decryption-with-leakage oracle on C with its final bit flipped. In the real world, where $C = \mathcal{E}_k(M||T)$, the leakage will then equal the first bit of M with probability 1. Yet in the ideal world, C is independent of M, so the leakage will equal the first bit of M with probability half. Thus, testing whether the decryption leakage equals the first bit of M leads to a distinguisher with a significant advantage. However, this does not invalidate IND–CPLA security of the encryption scheme since it is a new ciphertext.

The result of the above observation is that no fully generic composition result granting security in the presence of leakage is possible for schemes where decryption follows a DtV or D&V structure. This affects the NRS compositions A1–A4, A7–A12, N1, N3 and N4; none of which can be regarded as generically secure under leakage.

Less general composition results might still be possible, for instance by restricting the leakage classes of the primitives. After all, in the trivial case that the leakage classes are all $\emptyset$, the original NRS results hold directly. We leave open whether significantly larger realistic leakage classes exist leading to secure MtE constructions.

Alternatively, stronger assumptions on $\mathcal{E}$ could help. For instance, if $\mathcal{E}$'s security matches that of a tweakable (variable input length) cipher, the MAC-then-Encrypt constructions become a sort of encode-then-encipher. The latter is secure against release of unverified plaintext [23]. We leave open the identification of sufficient conditions on $\mathcal{E}$ for a generic composition result in the presence of leakage to pull through for EtM or E&M; relatedly, we leave open the extension of our work to the encode-then-encipher setting.

5.3 Encrypt-then-MAC is Secure under Leakage

The iv-based schemes A5 and A6, as well as the nonce-based N2, all fall under the EtM design. The inverse of an EtM scheme can be D&V or VtD, but due to the reasons discussed in the previous section it is only possible to achieve meaningful security if decryption obeys the VtD ordering. These schemes, along with the iv2n mechanism for building a nonce-based encryption scheme out of an iv-based one, are all represented in Figure 2. Before proving their security, we begin with some observations about EtM–VtD designs in the face of leakage.

Initial Observations. Since the final ciphertext will be formed from an encryption ciphertext and a tag, if the overall output is to be indistinguishable from random bits, then so must the tag. Thus we require both that $(\mathcal{T}, \mathcal{V})$ is a secure $(\mathcal{L}_\mathsf{T}, \mathcal{L}_\mathsf{V})$-LMAC, and that $\mathcal{T}$ is a secure $\mathcal{L}_\mathsf{T}$-LPRF. Shrimpton and Terashima [53] defined a (weaker) authenticated encryption notion where the 'recovery information' does not need to be random, only the ciphertext, in which case one may drop the second requirement.

In the traditional case, it is possible to build secure EtM schemes from an encryption scheme that is IND–CPA secure. This is because (by assumption on the security of the MAC), the only output the adversary can ever receive from the internal decryption function $\mathcal{D}$ is a plaintext corresponding to a previous $\mathcal{E}$ query. However, in the leakage setting, this no longer holds, because such a query also allows the adversary to evaluate a leakage function $L \in \mathcal{L}_\mathsf{D}$, albeit on a (N, C) pair that they already know. If $\mathcal{L}_\mathsf{D}$ contained functions that revealed sufficient information about the key, this would leave the composed scheme completely broken, despite being secure in the IND–CPLA game. The augmented IND–aCPLA game, in which the adversary is allowed to make these decryption queries, is sufficiently nuanced to prevent this.

Security of EtM Composition Schemes. We now describe the security of the composition schemes A5, A6 and N2, and the iv2n construction. Working under the assumption of OCLI-style leakage, as described in Section 5.1, we will reduce the security of the composition to the security of its components. Technically we also have a term quantifying any additional weaknesses due to the composition scheme, but in all cases this will be zero. The proofs can be found in Appendix A. We begin with N2, and show it is essentially as secure as the weakest of its components, by constructing explicit adversaries against each.

Theorem 1 (nLAE from nLE and LPRF via $\mathsf{N2}$ **composition).** *Let* $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}}, \mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$ *be leakage sets for the appropriate primitives, and define* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$ *as in Section 5.1. Let* $\mathbb{A}$ *be an adversary against the* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$*-nLAE security of* $\mathsf{N2}[\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}]$*. Then, there exist adversaries* $\mathbb{A}_{\mathrm{CPA}}, \mathbb{A}_{\mathrm{MAC}}, \mathbb{A}_{\mathrm{PRF}}$ *(with similar complexity to* $\mathbb{A}$*) against the* $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$*-nLE security of* $(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}})$*, the* $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$*-LMAC security of* $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ *and the* $\mathcal{L}_{\mathsf{T}}$*-LPRF security of* $\boldsymbol{\mathcal{T}}$ *respectively, such that:*

$$\mathbf{Adv}^{\mathrm{nLAE}}_{\mathsf{N2};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}) \leq \\ \mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{E}},\mathcal{L}_{\mathsf{D}}}(\mathbb{A}_{\mathrm{CPA}}) + \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{T}};\mathcal{L}_{\mathsf{T}}}(\mathbb{A}_{\mathrm{PRF}}) + \mathbf{Adv}^{\mathrm{EUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_{\mathsf{T}},\mathcal{L}_{\mathsf{V}}}(\mathbb{A}_{\mathrm{MAC}}) .$$

As the following result shows, the intuitive mechanism for building a nLE scheme from a secure ivLE scheme and a secure LPRF is itself secure. While unsurprising, this will allow us to instantiate the $\mathsf{N2}$ construction with the more common object of an ivLE scheme.

Theorem 2 (nLE from ivLE and LPRF). *Let* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}}, \mathcal{L}_{\mathsf{F}})$ *be leakage sets for the appropriate primitives, and define* $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$ *as in Section 5.1. Let* $\mathbb{A}$ *be an adversary against the* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}})$*-nLE security of* $\mathsf{iv2n}[\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{F}}]$*. Then, there exist adversaries* $\mathbb{A}_{\mathrm{CPA}}$, $\mathbb{A}_{\mathrm{PRF}}$ *(with similar complexity to* $\mathbb{A}$*) against the* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}})$*-ivLE security of* $(\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}})$*, and the* $\mathcal{L}_{\mathsf{F}}$*-LPRF security of* $\boldsymbol{\mathcal{F}}$ *respectively, such that:*

$$\mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\mathsf{iv2n};\mathcal{L}_{\mathsf{E}},\mathcal{L}_{\mathsf{D}}}(\mathbb{A}) \leq \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{F}};\mathcal{L}_{\mathsf{F}}}(\mathbb{A}_{\mathrm{PRF}}) + \mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\mathbf{iv}\boldsymbol{\mathcal{E}},\mathbf{iv}\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{ivE}},\mathcal{L}_{\mathsf{ivD}}}(\mathbb{A}_{\mathrm{CPA}}),$$

Pulling these two results together, we are able to prove the security of the $\mathsf{A5}$ construction. The security of $\mathsf{A6}$ against adversaries who never repeat the pair (N, A) can be easily recovered from this by considering it as an equivalent representation of the $\mathsf{A5}$ scheme acting on nonce space $\mathsf{N}' = \mathsf{N} \times \mathsf{A}$ but with no associated data.

Theorem 3 (nLAE from ivLE and LPRF via $\mathsf{A5}$ **composition).** *Let* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}}, \mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}}, \mathcal{L}_{\mathsf{F}})$ *be leakage sets for the appropriate primitives, and define* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$ *as in Section 5.1. Let* $\mathbb{A}$ *be an adversary against the* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$*-nLAE security of* $\mathsf{A5}[\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{F}}; \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}]$*. Then, there exist adversaries* $\mathbb{A}_{\mathrm{CPA}}, \mathbb{A}_{\mathrm{PRF}}, \mathbb{A}_{\mathrm{MAC}}, \mathbb{A}'_{\mathrm{PRF}}$ *(with similar complexity to* $\mathbb{A}$*) against the* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}})$*-ivLE security of* $(\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}})$*, the* $\mathcal{L}_{\mathsf{F}}$*-LPRF security of* $\boldsymbol{\mathcal{F}}$*, the* $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$*-LMAC security of* $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ *and the* $\mathcal{L}_{\mathsf{T}}$*-LPRF security of* $\boldsymbol{\mathcal{T}}$ *respectively, such that:*

$$\mathbf{Adv}^{\mathrm{nLAE}}_{\mathsf{A5};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}) \leq \mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{ivE}},\mathcal{L}_{\mathsf{ivD}}}(\mathbb{A}_{\mathrm{CPA}}) + \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{F}};\mathcal{L}_{\mathsf{F}}}(\mathbb{A}_{\mathrm{PRF}}) \\ + \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{T}};\mathcal{L}_{\mathsf{T}}}(\mathbb{A}'_{\mathrm{PRF}}) + \mathbf{Adv}^{\mathrm{sEUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_{\mathsf{T}},\mathcal{L}_{\mathsf{V}}}(\mathbb{A}_{\mathrm{MAC}}),$$

Theorem 4 (nLAE from ivLE and LPRF via $\mathsf{A6}$ **composition).** *Let* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}}, \mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}}, \mathcal{L}_{\mathsf{F}})$ *be leakage sets for the appropriate primitives, and define* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$ *as in Section 5.1. Let* $\mathbb{A}$ *be an adversary against the* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$*-LAE security of* $\mathsf{A6}[\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{F}}; \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}]$ *who does not make two encryption queries with the same* (N, A) *pair. Then, there exist adversaries* $\mathbb{A}_{\mathrm{CPA}}, \mathbb{A}_{\mathrm{PRF}}, \mathbb{A}_{\mathrm{MAC}}, \mathbb{A}'_{\mathrm{PRF}}$ *(with similar complexity to* $\mathbb{A}$*) against the* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}})$*-ivLE security of* $(\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}})$*, the* $\mathcal{L}_{\mathsf{F}}$*-LPRF security of* $\boldsymbol{\mathcal{F}}$*, the* $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$*-sEUF-CLMA security of* $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ *and the* $\mathcal{L}_{\mathsf{T}}$*-LPRF security of* $\boldsymbol{\mathcal{T}}$ *respectively, such that:*

$$\mathbf{Adv}^{\mathrm{nLAE}}_{\mathsf{A6};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}) \leq \mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{ivE}},\mathcal{L}_{\mathsf{ivD}}}(\mathbb{A}_{\mathrm{CPA}}) + \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{F}};\mathcal{L}_{\mathsf{F}}}(\mathbb{A}_{\mathrm{PRF}}) \\ + \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{T}};\mathcal{L}_{\mathsf{T}}}(\mathbb{A}'_{\mathrm{PRF}}) + \mathbf{Adv}^{\mathrm{sEUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_{\mathsf{T}},\mathcal{L}_{\mathsf{V}}}(\mathbb{A}_{\mathrm{MAC}}),$$

5.4 Achieving Misuse Resistant LAE security

In Section 5.2 we discussed why no composition scheme can be (generically) secure against leakage if it begins by calculating a candidate plaintext. This meant ruling out every NRS construction secure in the nonce misuse model, an important feature for a modern robust AE schemes [10, 23, 49]. Roughly

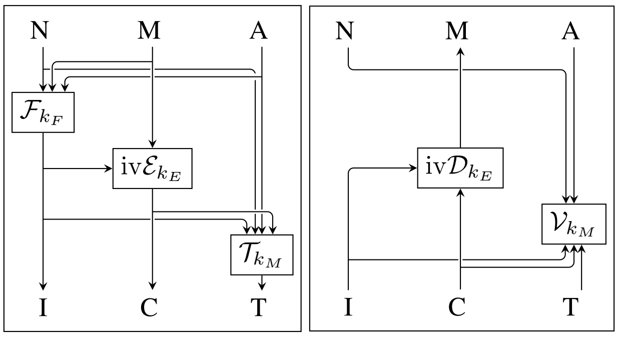

Fig. 7: The Synthetic-IV-and-Tag (SIVAT) scheme. On the left, the encryption routine runs from top to bottom, outputting a ciphertext $I||C||T$. Decryption (on the right) runs from bottom to top. If during decryption verification fails, and $\mathcal{V}_{\boldsymbol{k}_m}$ returns $\perp$, no further computations are performed. In the decryption direction, the PRF $\mathcal{F}$ is not required.

speaking, to be MRAE a scheme must be MtE (to ensure maximum diffusion) and to be leakage resilient it must be EtM (to ensure minimal leakage).

The Synthetic IV and Tag (SIVAT) scheme, defined in Figure 7, addresses the combined mrLAE goal, and can be seen as MtEtM. It be seen as composing the SIV construction [49] (referred to as A4 in NRS) with a secure MAC, or alternatively as the natural strengthening of A6 towards nonce misuse security, by adding the message to the IV calculation and making the appropriate modifications this leads to.

This additional feature does come at a cost. While schemes in the traditional setting achieve misuse resistance for the same ciphertext expansion as non-resistant schemes, the SIVAT scheme requires essentially double that. It also has a large number of internal wires, with each function taking in a large number of inputs, although removing any one leads to incorrectness or insecurity. For encryption calls, all inputs must go into the LPRF (for misuse resistance) and for decryption they must go into verification (to prevent RUP attacks).

The proof is very similar to that for A5 or A6 (Theorems 3 and 4), since the additional element of a SIVAT ciphertext (I) is present in those settings, and might already be available to the adversary through leakage.

Theorem 5 (mrLAE from ivLE and LPRF via SIVAT composition). *Let* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}}, \mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}}, \mathcal{L}_{\mathsf{F}})$ *be leakage sets for the appropriate primitives, and define* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$ *as in Section 5.1. Let* $\mathbb{A}$ *be an adversary against the* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$*-mrLAE security of* $\mathsf{SIVAT}[\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{F}}; \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}]$*. Then, there exist adversaries* $\mathbb{A}_{\mathrm{CPA}}$*,* $\mathbb{A}_{\mathrm{PRF}}$*,* $\mathbb{A}_{\mathrm{MAC}}$ *and* $\mathbb{A}'_{\mathrm{PRF}}$ *(with similar complexity to* $\mathbb{A}$*) against the* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}})$*-ivLE security of* $(\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}})$*, the* $\mathcal{L}_{\mathsf{F}}$*-LPRF security of* $\boldsymbol{\mathcal{F}}$*, the* $(\mathcal{L}_{\mathsf{T}}, \mathcal{L}_{\mathsf{V}})$*-EUF-CLMA security of* $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ *and the* $\mathcal{L}_{\mathsf{T}}$*-LPRF security of* $\boldsymbol{\mathcal{T}}$ *respectively, such that:*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{LAE}}_{\mathsf{SIVAT};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}) \leq{}& \mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{ivE}},\mathcal{L}_{\mathsf{ivD}}}(\mathbb{A}_{\mathrm{CPA}}) + \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{F}};\mathcal{L}_{\mathsf{F}}}(\mathbb{A}_{\mathrm{PRF}}) \\ &+ \mathbf{Adv}^{\mathrm{LPRF}}_{\boldsymbol{\mathcal{T}};\mathcal{L}_{\mathsf{T}}}(\mathbb{A}'_{\mathrm{PRF}}) + \mathbf{Adv}^{\mathrm{EUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_{\mathsf{T}},\mathcal{L}_{\mathsf{V}}}(\mathbb{A}_{\mathrm{MAC}}).\end{aligned}$$

6 A mrLAE scheme

The A5 and SIVAT composition mechanisms can be instantiated with any suitably secure primitives to yield secure nLAE or mrLAE schemes. In Appendix B, we provide an example of this, proven secure in the generic group model, and with leakage following the OCLI paradigm. A thorough discussion of the design choices, specifications, and security justification can be found in the relevant appendices, but for completeness we provide a brief descriptions here, along with the final theorem statement.

Our construction is bootstrapped from an alternative implementation of the MAC of Martin *et al.* [34]. We prove that, by using three shares rather than the original two, the tagging function is also a LPRF, and demonstrate how to achieve security against verification leakage. We show that CFB mode is a secure IND–aCPLA scheme when built around a LPRF that is suitably secure. Together, these allow us to construct a secure mrLAE scheme through the SIVAT mechanism.

Theorem 6. *Let* $(\mathcal{L}_{\mathsf{Enc}}, \mathcal{L}_{\mathsf{Dec}})$ *be the* SIVAT *mechanism instantiated with the implementations described in Appendix B, over a generic group of* p *elements, and assume that each share of the internal PRF leaks at most* λ *per call. Let* $\mathbb{A}$ *be an adversary making at most* g *direct queries to the generic group oracle,* h *queries to the hashing oracle, and* q *construction queries totalling* σ *blocks.*

$$\mathbf{Adv}^{\mathrm{LAE}}_{\mathsf{SIVAT};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}) \leq \frac{27 \cdot 2^{4\lambda} \cdot (g \cdot \sigma + 1) \cdot (9q + 5\sigma + g)^2}{p}.$$

To get a feel for the practical security level, let's look at parameters if the schemes are instantiated over a 512 bit elliptic curves, and we want the keep the attack success probability below 2^{-60} (a common limit in the real world, *e.g.* [32]). Let's assume that each internal leakage function leaks at most $\lambda = 85$ bits, which is approximately a sixth of a group element. Then the scheme would remain secure until the adversary has encrypted or decrypted 2^{26} blocks, and made a similar number of queries to the generic group. If Shrimpton and Terashima's [53] (weaker) notion were considered to be sufficient, where "recovery information" is not required to be random and hence the LMAC's tagging algorithm need not also be a LPRF, we would get even better bounds.

This result comes with a few caveats, covered in more detail by Appendix B.1. In the leakage-free setting, the scheme is secure in the random oracle model [9], and the leakage resilience security is proven in the generic group model [52]. Moreover, to ensure security against the leakage of arbitrary functions of the key, to process q queries of total σ blocks the construction must sample $4q + \sigma$ random group elements in a leakage-resilient manner, which can be complicated [34]. That said, the example provided is sufficient to demonstrate such schemes can (and indeed, do) exist: improving upon this result is an obvious candidate for future work.

7 Conclusions and Open Problems

We introduced notions for strengthened AE when considering leakage, discussed generic composition under leakage, and showed the EtM type constructions can be proven secure in this context. We give a new scheme, SIVAT, that achieves misuse resistance and leakage resilience simultaneously and give a concrete instantiation for it. Our research unveils several interesting open problems, which we summarise subsequently.

IND–aCPLA. We conjecture that for many reasonable leakage sets $\mathcal{L}_{\mathsf{E}}$, $\mathcal{L}_{\mathsf{E}}$-IND–CPLA implies $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{E}})$-IND–aCPLA, up to some minor bookkeeping to ensure correct syntax. We leave it as an interesting question to prove this or find a counter-example. More generally, is there some way of defining $\mathcal{L}_{\mathsf{ED}}$ as a function of some general sets $\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}}$ such that $\mathcal{L}_{\mathsf{ED}}$-IND–CPLA $\implies$ $(\mathcal{L}_{\mathsf{E}}, \mathcal{L}_{\mathsf{D}})$-IND–aCPLA. A trivial result exists if one allows nonce-reuse (since repeated $\mathcal{E}$-queries can simulate any valid $\mathcal{D}$-queries), but the nonce or iv respecting cases remain open.

MtE with restricted leakage sets. The insecurity of the majority of the MtE schemes when considering leakage comes from a generic attack against any schemes whose inverse follows the decrypt-then-verify or decrypt-and-verify structure. We leave it as an interesting open question to investigate the leakage security under other more restricted leakage sets.

Misuse resistance without message expansion. We demonstrate that misuse resistance can be achieved through generic composition, at the cost of additional message expansion, using a MAC-then-Encrypt-then-MAC structure (leading to SIVAT). We believe that dedicated constructions are likely to be exist that can achieve mrLAE security with minimal expansion, or more generally LAE without requiring independent keys.

References

1. Alkassar, A., Geraldy, A., Pfitzmann, B., Sadeghi, A.R.: Optimized self-synchronizing mode of operation. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 78–91. Springer, Heidelberg (Apr 2002); Cited on page 33.
2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: Sarkar and Iwata [50], pp. 105–125; Cited on pages 3, 4, 5, 6, 15, and 18.
3. Aranha, D.F., Fouque, P.A., Qian, C., Tibouchi, M., Zapalowicz, J.C.: Binary elligator squared. In: Joux, A., Youssef, A.M. (eds.) SAC 2014. LNCS, vol. 8781, pp. 20–37. Springer, Heidelberg (Aug 2014); Cited on page 30.
4. Barwell, G., Page, D., Stam, M.: Rogue decryption failures: Reconciling AE robustness notions. In: Groth [21], pp. 94–111 ; Cited on pages 3, 4, 5, 8, and 15.
5. Belaïd, S., Fouque, P.A., Gérard, B.: Side-channel analysis of multiplications in GF(2128) - application to AES-GCM. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 306–325. Springer, Heidelberg (Dec 2014) ; Cited on page 16.
6. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press (Oct 1997); Cited on page 30.
7. Bellare, M., Kane, D., Rogaway, P.: Big-key symmetric encryption: Resisting key exfiltration. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 373–402. Springer, Heidelberg (Aug 2016); Cited on page 5.
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 (Dec 2000) ; Cited on pages 3 and 5.
9. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993); Cited on pages 22 and 30.
10. Bernstein, D.J.: CAESAR competition call (2013), `http://competitions.cr.yp.to/caesar-call-3.html`; Cited on pages 5 and 20.
11. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13. pp. 967–980. ACM Press (Nov 2013) ; Cited on page 30.
12. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 367–390. Springer, Heidelberg (Mar 2014) ; Cited on pages 3, 4, 5, and 15.
13. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: Security of symmetric encryption in the presence of ciphertext fragmentation. Cryptology ePrint Archive, Report 2015/059 (2015), `http://eprint.iacr.org/2015/059`; Cited on page 3.
14. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (May 2005); Cited on page 30.
15. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 621–630. ACM Press (May / Jun 2009); Cited on pages 3, 4, and 16.
16. Dodis, Y., Pietrzak, K.: Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (Aug 2010); Cited on pages 6, 14, and 30.
17. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th FOCS. pp. 293–302. IEEE Computer Society Press (Oct 2008); Cited on pages 3, 4, and 6.
18. Faust, S., Pietrzak, K., Schipper, J.: Practical leakage-resilient symmetric cryptography. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 213–232. Springer, Heidelberg (Sep 2012); Cited on pages 6, 14, and 30.
19. Fischlin, M., Günther, F., Marson, G.A., Paterson, K.G.: Data is a stream: Security of stream-based channels. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 545–564. Springer, Heidelberg (Aug 2015) ; Cited on page 3.
20. Galindo, D., Vivek, S.: A practical leakage-resilient signature scheme in the generic group model. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 50–65. Springer, Heidelberg (Aug 2013); Cited on page 30.
21. Groth, J. (ed.): 15th IMA International Conference on Cryptography and Coding, LNCS, vol. 9496. Springer, Heidelberg (Dec 2015); Cited on pages 23 and 24.
22. Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-resilient cryptography from minimal assumptions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 160–176. Springer, Heidelberg (May 2013) ; Cited on pages 5 and 16.
23. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 15–44. Springer, Heidelberg (Apr 2015) ; Cited on pages 3, 5, 19, and 20.
24. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (Aug 2003); Cited on pages 4 and 16.
25. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC (2008); Cited on page 7.
26. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (Dec 2009); Cited on page 16.
27. Kiltz, E., Pietrzak, K.: Leakage resilient ElGamal encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer, Heidelberg (Dec 2010); Cited on pages 3, 4, 29, and 30.

28. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (Aug 1996); Cited on page 3.
29. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (Aug 1999); Cited on page 3.
30. Kurosawa, K., Phong, L.T.: Leakage resilient IBE and IPE under the DLIN assumption. In: Jacobson Jr., M.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 13. LNCS, vol. 7954, pp. 487–501. Springer, Heidelberg (Jun 2013) ; Cited on page 16.
31. Longo, J., Martin, D.P., Oswald, E., Page, D., Stam, M., Tunstall, M.: Simulatable leakage: Analysis, pitfalls, and new constructions. In: Sarkar and Iwata [50], pp. 223–242; Cited on page 6.
32. Luykx, A., Paterson, K.: Limits on authenticated encryption use in tls (2016), http://www.isg.rhul.ac.uk/~kp/TLS-AEbounds.pdf; Cited on page 22.
33. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer (2008); Cited on page 11.
34. Martin, D.P., Oswald, E., Stam, M., Wójcik, M.: A leakage resilient MAC. In: Groth [21], pp. 295–310; Cited on pages 4, 6, 14, 21, 22, 29, and 34.
35. Maurer, U.M.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (Dec 2005) ; Cited on page 30.
36. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode of operation (full version). Cryptology ePrint Archive, Report 2004/193 (2004), http://eprint.iacr.org/2004/193; Cited on page 16.
37. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (Feb 2004); Cited on pages 3, 4, 16, and 18.
38. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 257–274. Springer, Heidelberg (May 2014); Cited on pages 3, 4, 5, and 10.
39. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes 55(2), 165–172 (1994); Cited on page 30.
40. NIST: FIPS 81: DES Modes of Operation. Issued December 2, 63 (1980); Cited on page 3.
41. Pereira, O., Standaert, F.X., Vivek, S.: Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In: Ray, I., Li, N., Kruegel:, C. (eds.) ACM CCS 15. pp. 96–108. ACM Press (Oct 2015); Cited on page 6.
42. Perrin, T.: Double ratchet algorithm (2014), https://github.com/trevp/double_ratchet/wiki, Retrieved 2016-09-01; Cited on page 6.
43. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (Apr 2009); Cited on page 6.
44. Qin, B., Liu, S.: Leakage-flexible CCA-secure public-key encryption: Simple construction and free of pairing. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 19–36. Springer, Heidelberg (Mar 2014); Cited on page 5.
45. Renauld, M., Standaert, F.X., Veyrat-Charvillon, N., Kamel, D., Flandre, D.: A formal study of power variability issues and side-channel attacks for nanoscale devices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 109–128. Springer, Heidelberg (May 2011); Cited on page 18.
46. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 02. pp. 98–107. ACM Press (Nov 2002); Cited on page 7.
47. Rogaway, P.: Nonce-based symmetric encryption. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 348–359. Springer, Heidelberg (Feb 2004); Cited on page 7.
48. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS 01. pp. 196–205. ACM Press (Nov 2001); Cited on page 7.
49. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (May / Jun 2006); Cited on pages 3, 5, 20, and 21.
50. Sarkar, P., Iwata, T. (eds.): ASIACRYPT 2014, Part I, LNCS, vol. 8873. Springer, Heidelberg (Dec 2014); Cited on pages 23 and 24.
51. Schipper, J.: Leakage-Resilient Authentication. Ph.D. thesis, Utrecht University (2010); Cited on page 6.
52. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (May 1997); Cited on pages 22 and 30.
53. Shrimpton, T., Terashima, R.S.: A modular framework for building variable-input-length tweakable ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 405–423. Springer, Heidelberg (Dec 2013) ; Cited on pages 19, 22, and 34.
54. Standaert, F.X., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 335–352. Springer, Heidelberg (Aug 2013) ; Cited on pages 3, 4, and 15.
55. Tibouchi, M.: Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 139–156. Springer, Heidelberg (Mar 2014); Cited on page 30.
56. Yau, A.K.L., Paterson, K.G., Mitchell, C.J.: Padding oracle attacks on CBC-mode encryption with secret and random IVs. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 299–319. Springer, Heidelberg (Feb 2005) ; Cited on page 3.
57. Yu, Y., Standaert, F.X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 10. pp. 141–151. ACM Press (Oct 2010); Cited on page 11.

A Security Proof for Generic Composition

Proof (Of Theorem 1). We will split into two cases, based on whether the adversary constructs a forgery against the MAC or not. The first of these will be dealt with by constructing an adversary $\mathbb{A}_{\text{MAC}}$ against the LMAC security of the MAC. Then, under the assumption that this does not happen, we reduce to the IND–aCPLA security of $\boldsymbol{\mathcal{E}}$, before finally using the PRLF security of $\boldsymbol{\mathcal{T}}$ to complete the proof. This is formalised using the oracles of Figure 8.

Using an identical-until-bad argument with variable forge and the event that forge $\to$ true, we first simplify the real and ideal worlds.

$$\begin{aligned}&\Big|\Pr\left[\mathbb{A}^{\mathsf{Enc}_k,\mathsf{Dec}_k,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k}\to 1 \mid \mathsf{forge}\right]\cdot\Pr\left[\mathbb{A}^{\mathsf{Enc}_k,\mathsf{Dec}_k,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k}\ \mathsf{sets\ forge}\right]\\ &-\Pr\left[\mathbb{A}^{\$,\perp,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k}\to 1 \mid \mathsf{forge}\right]\cdot\Pr\left[\mathbb{A}^{\$,\perp,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k}\ \mathsf{sets\ forge}\right]\Big|\\ &\quad\le\max(\Pr\left[\mathbb{A}^{\mathsf{Enc}_k,\mathsf{Dec}_k,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k}\ \mathsf{sets\ forge}\right],\Pr\left[\mathbb{A}^{\$,\perp,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k}\ \mathsf{sets\ forge}\right]).\end{aligned}$$

Until forge occurs, $\mathcal{D}_k$ is identical to $\perp$, because the adversary may not forward queries from $\ell[\mathbf{Enc}]_k$ to their challenge decryption oracle (or this would provision a trivial win). So, the only difference between these two terms is whether the first oracle is honest (Enc_k) or ideal (\$). We cannot in general determine which of these terms is larger since for any $\mathbb{A}$ there exists an $\mathbb{B}$ who simply runs $\mathbb{A}$ and inverts the answer of $\mathbb{A}$ as his own.

We bound these by constructing an adversary $\mathbb{A}^f_{\text{MAC}}$ against the LMAC security of $(\mathcal{T},\mathcal{V})$, where $f\in\{\mathcal{E}_k,\$\}$ defines which of these two worlds $\mathbb{A}^f_{\text{MAC}}$ will simulate. $\mathbb{A}^f_{\text{MAC}}$ begins by sampling a key for the encryption scheme $(\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{D}})$ and instantiates her internal f oracle as either $\mathcal{E}_k$ or a random function \$ as required. She runs $\mathbb{A}$ on an instantiation of the N2 construction, and forwarding any LMAC related queries to her own oracles. For challenge encryption queries, she answers with f, and any other encryption-related queries are answered by her own version of $\boldsymbol{\mathcal{E}}$. If $\mathbb{A}$ completes without triggering forge, they return 0. If $\mathbb{A}$ triggers forge through a $\ell[\mathbf{Dec}]_k$ query, $\mathbb{A}^f_{\text{MAC}}$ resets forge and repeats the query to $\mathcal{D}$. Then, $\mathbb{A}^f_{\text{MAC}}$ returns 1 if the $\mathcal{D}$ set forge. This can only happen if $\mathcal{V}$ returned $\top$, meaning they have distinguished the honest verification. Since $\mathcal{L}_{\mathsf{Enc}}$ and $\mathcal{L}_{\mathsf{Dec}}$ obey the OCLI assumption, at no point does $\mathbb{A}$ ask a leakage query that it is impossible for $\mathbb{A}^f_{\text{MAC}}$ to answer, and at no point does $\mathbb{A}$ ask any other query $\mathbb{A}^f_{\text{MAC}}$ forbidden from asking. Thus $\mathbb{A}^f_{\text{MAC}}$ is always able to run $\mathbb{A}$ in this manner, and so

$$\Pr\left[\mathbb{A}^{f,\mathsf{Dec}_k,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k}\ \mathsf{sets\ forge}\right]=\mathbf{Adv}^{\mathrm{EUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_{\mathsf{T}},\mathcal{L}_{\mathsf{V}}}(\mathbb{A}^f_{\text{MAC}}).$$

Since we are interested in simply an existence result, define $\mathbb{A}_{\text{MAC}}$ to be whichever of $\mathbb{A}^{\$}_{\text{MAC}}$ and $\mathbb{A}^{\mathcal{E}_k}_{\text{MAC}}$ maximises this term. So,

$$\begin{aligned}\mathbf{Adv}^{\mathrm{LAE}}_{\mathsf{N2};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}):=&\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathsf{Enc}_k,\mathsf{Dec}_k,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k\\ \$\ \ ,\ \ \perp\ \ ,\ell[\mathbf{Enc}]_k,\ell[\mathbf{Dec}]_k\end{pmatrix}\\ &\le\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathsf{Enc}_k,\boxed{\mathsf{Dec}_k},\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\\ \$\ \ ,\ \ \perp\ \ ,\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\end{pmatrix}+\mathbf{Adv}^{\mathrm{EUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_{\mathsf{T}},\mathcal{L}_{\mathsf{V}}}(\mathbb{A}_{\text{MAC}}).\end{aligned}$$

So, consider now the remaining advantage term. Since the adversary is prohibited from passing queries from his honest $\ell[\mathbf{Enc}]_k$ oracle to his challenge decryption oracle, $\boxed{\mathsf{Dec}_k}$ is identical to $\perp$. Thus we are left just to bound the term

$$\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathsf{Enc}_k,\perp,\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\\ \$\ \ ,\perp,\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\end{pmatrix}\le\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathsf{Enc}_k\ ,\perp,\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\\ \boxed{\mathsf{Enc}_k},\perp,\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\end{pmatrix}+\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\boxed{\mathsf{Enc}_k},\perp,\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\\ \$\ \ ,\perp,\ell[\mathbf{Enc}]_k,\boxed{\ell[\mathbf{Dec}]_k}\end{pmatrix}. \tag{1}$$

Moreover, $\boxed{\ell[\mathbf{Dec}]_k}$ can be simplified slightly, since the final else section is entered if and only if $(N,A,C)\in\mathcal{X}$. That is, the adversary is only provided with leakage on the internal decryption function

```
function Enc_k(N, A, M)
    $k_e||k_m \leftarrow k$
    $C_e \leftarrow \mathcal{E}(k_e, N, M)$
    $\boxed{C_e \leftarrow \$(N, M)}$
    $T \leftarrow \mathcal{T}(k_m, N||A||C_e)$
    $C \leftarrow C_e||T$
    return $C$
```

```
function Dec_k(N, A, C)
    $k_e||k_m \leftarrow k$
    $C_e||T \leftarrow C$
    $v \leftarrow \mathcal{V}(k_m, N||A||C_e, T)$
    if $(N, A, C) \notin \mathcal{X} \wedge v = \top$ then
        forge ← true
        $\boxed{v \leftarrow \bot}$
    if $v = \bot$ then
        return $\bot$
    $M \leftarrow \mathcal{D}(k_e, N, A, C_e)$
    return $M$
```

```
function $\ell[\mathsf{Enc}]_k(N, A, M; L)$
    $\boldsymbol{k}_e||\boldsymbol{k}_m \leftarrow \boldsymbol{k}$
    $L_e||L_t \leftarrow L$
    $r_e||r_m \leftarrow\$ \mathsf{R}$
    $C_e, \boldsymbol{k}'_e \leftarrow \mathcal{E}(\boldsymbol{k}_e, N, M; r_e)$
    $\Lambda_e \leftarrow L_e(\boldsymbol{k}_e, N, M; r_e)$
    $T, \boldsymbol{k}'_m \leftarrow \mathcal{T}(\boldsymbol{k}_m, N||A||C_e; r_m)$
    $\Lambda_t \leftarrow L_t(\boldsymbol{k}_m, N||A||C_e; r_m)$
    $\boldsymbol{k} \leftarrow \boldsymbol{k}'_e||\boldsymbol{k}'_m$
    $C \leftarrow C_e||T$
    $\mathcal{X} \xleftarrow{\cup} (N, A, C)$
    return $(C, \Lambda_e||\Lambda_t)$
```

```
function $\ell[\mathsf{Dec}]_k(\boldsymbol{k}, N, A, C; L)$
    $\boldsymbol{k}_e||\boldsymbol{k}_m \leftarrow \boldsymbol{k}$
    $L_v||L_d \leftarrow L$
    $r_e||r_m \leftarrow\$ \mathsf{R}$
    $C_e||T \leftarrow C$
    $v, \boldsymbol{k}'_m \leftarrow \mathcal{V}(\boldsymbol{k}_m, N||A||C_e, T; r_m)$
    $\Lambda_v \leftarrow L_v(\boldsymbol{k}_m, N||A||C_e, T; r_m)$
    if $(N, A, C) \notin \mathcal{X} \wedge v = \top$ then
        forge ← true
        $\boxed{v \leftarrow \bot}$
    if $v = \bot$ then
        $\boldsymbol{k} \leftarrow \boldsymbol{k}_e||\boldsymbol{k}'_m$
        return $(\bot, \Lambda_v)$
    else
        $M, \boldsymbol{k}'_e \leftarrow \boldsymbol{\mathcal{D}}(\boldsymbol{k}_e, N, A, C_e; r_e)$
        $\Lambda_e \leftarrow L_d(\boldsymbol{k}_e, N, A, C_e; r_e)$
        $\boldsymbol{k} \leftarrow \boldsymbol{k}'_e||\boldsymbol{k}'_m$
        return $(M, \Lambda_v||\Lambda_d)$
```

Fig. 8: The oracles used in the proof of security for composition scheme $\mathsf{N2}[\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}]$. The oracles Enc_k, Dec_k implement the scheme honestly and do not include the boxed code. Oracles $\ell[\mathsf{Enc}]_k$, $\ell[\mathsf{Dec}]_k$ implement the leakage oracle and split the leakage function following the OCLI paradigm, again not including the boxed code. In each case, including the boxed code leads to a variant used in the proof, which we denote with a box. Thus $\boxed{\mathsf{Enc}_k}$ is Enc_k with the boxed code included.

$\boldsymbol{\mathcal{D}}$ if making a query equivalent to a previous encryption query. Define $\mathbb{A}_{\mathrm{CPA}}$ to be an IND–aCPLA adversary, interacting with $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}$. They will run $\mathbb{A}$ on a simulation of $\mathsf{N2}$, where the compositions encryption queries are forwarded to their own oracles, and the tagging queries are run locally by picking a fresh key and verification queries are always $\perp$. If $\mathbb{A}_{\mathrm{CPA}}$ is in the real world and their challenge oracle is honest, the oracles they provide are precisely $\mathsf{Enc}_k, \perp, \ell[\mathbf{Enc}]_k, \boxed{\ell[\mathbf{Dec}]_k}$. If they are in the ideal world, then the first oracle provided to $\mathbb{A}$ is instead $\boxed{\mathsf{Enc}_k}$. Thus they distinguish if and only if $\mathbb{A}$ does, and so

$$\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathsf{Enc}_k\ ,\perp, \ell[\mathbf{Enc}]_k, \boxed{\ell[\mathbf{Dec}]_k}\\ \boxed{\mathsf{Enc}_k},\perp, \ell[\mathbf{Enc}]_k, \boxed{\ell[\mathbf{Dec}]_k}\end{pmatrix} = \mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{E}},\mathcal{L}_{\mathsf{D}}}(\mathbb{A}_{\mathrm{CPA}}).$$

The only difference between $\$$ and $\boxed{\mathsf{Enc}_k}$ is that in the second case the tag is not sampled uniformly at random. We construction a PRLF adversary $\mathbb{A}_{\mathrm{PRF}}$ against the PRLF security of $\mathcal{T}$ in the obvious manner. They run $\mathbb{A}$ on a simulated version of $\mathsf{N2}$ and forward $\mathbb{A}$s result as their own. To answer oracle queries from $\mathbb{A}$, $\mathbb{A}_{\mathrm{PRF}}$ samples a key and uses this to simulate all the encryption-related functionality. Queries made to $\mathcal{T}$ via the Enc_k oracle are answered by their own challenge oracle, and queries made through $\ell[\mathbf{Enc}]_k$ by their own leakage oracle. Thus

$$\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\boxed{\mathsf{Enc}_k},\perp, \ell[\mathbf{Enc}]_k, \boxed{\ell[\mathbf{Dec}]_k}\\ \$\ \ ,\perp, \ell[\mathbf{Enc}]_k, \boxed{\ell[\mathbf{Dec}]_k}\end{pmatrix} = \mathbf{Adv}^{\mathrm{PRLF}}_{\mathcal{T};\mathcal{L}_{\mathsf{T}}}(\mathbb{A}_{\mathrm{PRF}}).$$

Collecting these bounds together gives the claimed result. □

Proof (Of Theorem 2). By triangle inequality,

$$\begin{aligned}\mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\mathsf{iv2n};\mathcal{L}_{\mathsf{E}},\mathcal{L}_{\mathsf{D}}}(\mathbb{A}) &= \Delta_{\mathbb{A}}\begin{pmatrix}\mathcal{E}_k,\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\\ \$\ ,\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\end{pmatrix}\\ &\le \Delta_{\mathbb{A}}\begin{pmatrix}\mathcal{E}_k\ ,\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\\ \boxed{\mathcal{E}_k},\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\end{pmatrix} + \Delta_{\mathbb{A}}\begin{pmatrix}\boxed{\mathcal{E}_k},\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\\ \$\ ,\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\end{pmatrix}.\end{aligned}$$

Let $\mathbb{A}_{\mathrm{PRF}}$ be a PRLF adversary against $\boldsymbol{\mathcal{F}}$. She runs $\mathbb{A}$ on a simulated version of the construction. To answer $\mathbb{A}$s challenge queries, they use their challenge oracle for $\boldsymbol{\mathcal{F}}$ and a random key to instantiate $\mathbf{iv}\boldsymbol{\mathcal{E}}$. Since the leakage sets follow the OCLI paradigm, any leakage function L_f is a valid query to make to their own leakage oracle, so the first half of the leakage oracle can be simulated using their own leakage oracles. For the second half of the leakage oracles, they use their $\mathbf{iv}\boldsymbol{\mathcal{E}}$ key to evaluate $\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}$ and L_e. Finally, they forward $\mathbb{A}$s answer as their own. If their challenge oracle is honest, this perfectly matches $(\mathcal{E}_k, \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k)$, whereas if their challenge oracle is ideal this exactly matches $(\boxed{\mathcal{E}_k}, \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k)$. Thus,

$$\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\mathcal{E}_k\ ,\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\\ \boxed{\mathcal{E}_k},\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\end{pmatrix} = \mathbf{Adv}^{\mathrm{PRLF}}_{\boldsymbol{\mathcal{F}};\mathcal{L}_{\mathsf{F}}}(\mathbb{A}_{\mathrm{PRF}})$$

Conversely, let $\mathbb{A}_{\mathrm{CPA}}$ be an adversary against the $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}})$-ivLE security of $(\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}})$. He runs $\mathbb{A}$, answering any encryption-related queries with his own oracles, and evaluating the rest locally. Then, he forwards $\mathbb{A}$s response as his own. At no point does $\mathbb{A}$ ask a query that he is himself forbidden from asking his oracles, and the leakage sets $(\ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k)$ follow the OCLI paradigm, meaning any leakage queries he forwards are valid. Moreover, since the variable I is randomly sampled, he is an iv-respecting adversary. The fact $\mathbb{A}$ has access to I, the IV, through leakage does not prevent this since it is only available after the fact, and not in advance. In the real world, the oracles he proves $\mathbb{A}$ match $\boxed{\mathcal{E}_k}, \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k$, and in the ideal case they match $\$, \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k$. Thus,

$$\underset{\mathbb{A}}{\Delta}\begin{pmatrix}\boxed{\mathcal{E}_k}, \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\\ \$\ , \ell[\boldsymbol{\mathcal{E}}]_k, \ell[\boldsymbol{\mathcal{D}}]_k\end{pmatrix} = \mathbf{Adv}^{\mathrm{IND-CPLA}}_{\mathbf{iv}\boldsymbol{\mathcal{E}},\mathbf{iv}\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{ivE}},\mathcal{L}_{\mathsf{ivD}}}(\mathbb{A}_{\mathrm{CPA}})$$

Substituting these into the above expansion competes the proof. □

```
function $\mathcal{E}_k(N, M)$
    $k_f||k_e \leftarrow k$
    $I \leftarrow \mathcal{F}(k_f, N, M)$
    $\boxed{I \leftarrow \$(N, M)}$
    $C \leftarrow \mathbf{iv}\boldsymbol{\mathcal{E}}(k_e, I, M)$
    return $C$
```

```
function $\ell[\boldsymbol{\mathcal{E}}]_k(N, M; L)$
    $\boldsymbol{k}_f||\boldsymbol{k}_e \leftarrow \boldsymbol{k}$
    $L_f||L_e \leftarrow L$
    $r_f||r_e \leftarrow\$ R$
    $I, \boldsymbol{k}'_f \leftarrow \mathcal{F}(\boldsymbol{k}_f, N, M; r_f)$
    $\Lambda_f \leftarrow L_f(\boldsymbol{k}_f, N, M; r_f)$
    $C, \boldsymbol{k}'_e \leftarrow \mathbf{iv}\boldsymbol{\mathcal{E}}(\boldsymbol{k}_e, I, M; r_e)$
    $\Lambda_e \leftarrow L_e(\boldsymbol{k}_e, I, M; r_e)$
    $\boldsymbol{k} \leftarrow \boldsymbol{k}'_f||\boldsymbol{k}'_e$
    return $(C, \Lambda_f||\Lambda_e)$
```

```
function $\ell[\boldsymbol{\mathcal{D}}]_k(N, M; L)$
    $\boldsymbol{k}_f||\boldsymbol{k}_e \leftarrow \boldsymbol{k}$
    $L_f||L_d \leftarrow L$
    $r_f||r_e \leftarrow\$ R$
    $I, \boldsymbol{k}'_f \leftarrow \mathcal{F}(\boldsymbol{k}_f, N, M; r_f)$
    $\Lambda_f \leftarrow L_f(\boldsymbol{k}, N, M; r_f)$
    $C, \boldsymbol{k}'_e \leftarrow \mathbf{iv}\boldsymbol{\mathcal{D}}(\boldsymbol{k}_e, I, M; r_e)$
    $\Lambda_d \leftarrow L_d(\boldsymbol{k}_e, I, M; r_e)$
    $\boldsymbol{k} \leftarrow \boldsymbol{k}'_f||\boldsymbol{k}'_e$
    return $(C, \Lambda_f||\Lambda_d)$
```

Fig. 9: Oracles from the security proof of iv2n. On the left $\mathcal{E}_k$ describes the scheme, and does not include the boxed code. $\boxed{\mathcal{E}_k}$, which does, is an intermediate step used in the proof. The centre and right are $\ell[\boldsymbol{\mathcal{E}}]_k$ and $\ell[\boldsymbol{\mathcal{D}}]_k$, with the leakage function and implementation written out in full as per their definitions. The leakage from $\ell[\boldsymbol{\mathcal{E}}]_k$ is $L_f||V||L_e$, and leakage from $\ell[\boldsymbol{\mathcal{D}}]_k$ is $L_f||V||L_d$.

Proof (Of Theorem 3). Chain Theorems 1 and 2. □

Proof (Of Theorem 5). We immediately observe that the SIVAT construction contains no wires that are not given to the adversary as output, or themselves inputs. Thus the leakage can be expressed very succinctly as leakage of the internal primitives alone, (along with the order in which this occurs). With this in mind, the proof proceeds in a similar, manner to that of Theorems 1 and 2 above, if anything being simpler. Let $\mathbb{A}$ be an adversary against the mrLAE security of SIVAT, and recall that this means he never repeats the triple (N, A, M) on an encryption query, nor forwards queries to/from his challenge oracles.

We begin almost identically to the proof of Theorem 1, performing an identical-until-bad switch on both real and ideal worlds. We define the event forge as whether the adversary can make a decryption query $(N, A, I||C_e||T)$ that is not the result of a previous encryption query but that $\mathcal{V}_{k_m}(N, A, I||C_e||T) = \top$ when interacting with oracles $(f, \mathcal{D}, \ell[\mathsf{Enc}]_k, \ell[\mathsf{Dec}]_k)$ for $f \in \{\mathsf{Enc}, \$\}$. By the same logic as before exist explicit adversaries $\mathbb{A}^f_{\mathrm{MAC}}$ such that the probability the adversary $\mathbb{A}$ triggering forge in either setting is less than that of $\mathbb{A}_{\mathrm{MAC}}$ winning the EUF–CMLA game.

Next, define an adversary against the PRLF security of $\boldsymbol{\mathcal{F}}$, analogous to that in 2. They build the SIVAT construction around their own challenge and leakage oracles, instantiated the ivE and tagging schemes with internally sampled primitives: after the previous switch verification can be idealised to reject all non-forwarded queries. They run $\mathbb{A}$ on this, and output $\mathbb{A}$s result as their own. Since $\mathbb{A}$ never repeats a triple (N, A, M), every query made to the PRF is unique, making $\mathbb{A}_{\mathrm{PRF}}$ a valid PRF adversary. So, $\mathbb{A}$ can distinguish whether $\mathcal{F}$ has been replaced with an idealised version if and only if $\mathbb{A}_{\mathrm{PRF}}$ wins the PRLF game.

Third, we replace $\mathcal{E}$ with the idealised form \$. As with Theorem 1, we construct the adversary $\mathbb{A}_{\mathrm{CPA}}$ against the IND–aCPLA security of $(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}})$ who builds this scheme by choosing the other primitives himself, runs $\mathbb{A}$ on it and forwards $\mathbb{A}$s result as his own. He is able to do this because the only decryption queries that call $\mathcal{D}$ or $\ell[\boldsymbol{\mathcal{D}}]_k$ are those which are repeats of previous queries, due to the earlier switch of $\mathcal{V}$. Then $\mathbb{A}$ distinguishes between these cases if and only if $\mathbb{A}_{\mathrm{CPA}}$ wins the IND–aCPLA game.

Finally, we construct an adversary $\mathbb{A}'_{\mathrm{PRF}}$ against the PRLF security of $\boldsymbol{\mathcal{T}}$, who simply runs $\mathbb{A}$ on an implementation of SIVAT in which every component has been idealised other than $\mathcal{T}$ and its associated leakage. This completes a chain of game hops which have taken us from the real to the ideal world, leaving just to collect the terms to form the final bound. □

B An instantiation of the SIVAT mechanism

In this section we describe how one can instantiate our generic scheme with concrete functions, again within the OCLI paradigm. At the lowest level, we construct PRLF and LMAC implementations where each algorithm consists of three phases. These implementations are secure as long as the phases leak

```
function $\mathcal{T}_k(M)$
    $Y_i \leftarrow e(k, H(M))$
    return $Y_i$

function $\mathcal{V}_k(M, T; r_{i+1})$
    $Y \leftarrow e(k, H(M))$
    return $(Y = T)$

function $\boldsymbol{\mathcal{T}}_2(M; r_{i+1})$
    $S^{◓}_{i+1}, W, Y_i^{◓} \leftarrow \mathcal{T}^{◓}(S_i^{◓}, M; r_{i+1})$
    $S^{◒}_{i+1}, Y_i \leftarrow \mathcal{T}^{◒}(S_i^{◒}, W, Y_i^{◓}; r_{i+1})$
    return $Y_i$
```

```
function $\mathcal{T}_2^{◓}(S_i^{◓}, M; r_{i+1})$
    $W \leftarrow H(M)$
    $Y_i^{◓} \leftarrow e(S_i^{◓}, W)$
    $S_{i+1}^{◓} \leftarrow S_i^{◓} \cdot r_{i+1}$
    return $(S_{i+1}^{◓}, W, Y_i^{◓})$

function $\mathcal{T}_2^{◒}(S_i^{◒}, W, Y_i^{◓}; r_{i+1})$
    $Y_i^{◒} \leftarrow e(S_i^{◒}, W)$
    $Y_i \leftarrow Y_i^{◓} \cdot Y_i^{◒}$
    $S_{i+1}^{◒} \leftarrow S_i^{◒} \cdot r_{i+1}^{-1}$
    return $(S_{i+1}^{◒}, Y_i)$
```

Fig. 10: The MOSW MAC Construction [34]. On the left are the tagging and verification functions $\mathcal{T}_k$ and $\mathcal{V}_k$. Below them is $\boldsymbol{\mathcal{T}}_2$, a secure $(\mathcal{L}_{\mathsf{T}_2}, \emptyset)$-LMAC implementation of the tagging function. The key and computation are split into two shares shown on the right, which can be assumed to leaks independently.

independently, and no phase leaks more than λ bits. On top of this PRLF, we construct an ivE scheme secure against arbitrary leakage from the composition and the same PRLF leakage as above.

We begin by restating the MAC of Martin *et al.* [34], on which our constructions are based. We prove that using three shares, the tagging implementation is a PRLF (but two shares are not sufficient). Next, we enhance the MAC by adjusting the verification routine such that it remains secure under leakage. Then, we show that a PRLF in CFB mode is a secure (iv-based) leakage resilient encryption scheme. Finally we combine these primitives within the A5 and SIVAT structures to form LAE and mrLAE schemes.

B.1 Recalling the MOSW MAC

Martin *et al.* [34] designed a MAC that computes the pairing of the key with a hash of the message. Using a technique of Kiltz and Pietrzak [27], they demonstrate how to split the tagging oracle into two shares. We will refer to it as the MOSW construction and this tagging implementation as $\boldsymbol{\mathcal{T}}_2$. It has been reproduced in Figure 10, using slightly different notation to the original.

They prove it secure against adaptive leakage on tagging queries from the leakage set $\mathcal{L}_{\mathsf{T}_2}$, the set containing all functions of the form

$$L_{T_2}((S^{◓}, S^{◒}), M; r) = (L^{◓}(S^{◓}, M; r), L^{◒}(S^{◒}, H(M), Y^{◓}; r))$$

where $L^{◓}$ and $L^{◒}$ output λ bits, $S^{◓}$ and $S^{◒}$ is the secret information used by each half, r is the randomness, and $Y^{◓}$ is the information passed from the first half of the function to the second half. It will form the basis for our constructions, but is not itself suitable for any of our requirements, since the authors do not provide an implementation secure under verification leakage, nor is it a PRLF (in the next section we will show that it is a PRF, but not a PRLF).

A note on notation. The symbols ◓, ◒ and later ⊖ will be used to refer to the different shares and subroutines used within the function. They are run sequentially from “top to bottom”.

In the MOSW work, $\mathcal{T}_2^{◓}$ generated the randomness it required and passed it to $\mathcal{T}_2^{◒}$. Here, we have $\boldsymbol{\mathcal{T}}_2$ generate all randomness and pass it into the functions. This is to match our definition of an implementation and keep the functions themselves more concise. However, as discussed in the MOSW work, how randomness is generated effects the amount of information that an adversary can extract from a device, and thus it must still be taken into consideration. Instead of giving the leakage function access to W, we write $H(M)$, to show how the input to the leakage function relates to the inputs of the functions.

B.2 Modelling Discussion

Before giving the details of the primitives used to instantiate the generic constructions from Sect. 5, we describe any assumptions that will be required for the remainder of this work. We explicitly detail any subtleties that arise from the interaction of leakage and well studied models.

Model assumptions. In the leakage free case, our constructions are secure in the random oracle model [9]. When leakage is available, we will extend to the generic group model, allowing us to model internal curve points. This is in contrast to the other works in the literature [20,27] that require the generic group model to achieve security even without leakage.

The generic group model [35,39,52] was designed to model groups where the adversary cannot exploit the structure of the underlying group. The model we follow here, due to Shoup [52], achieves this by representing group elements as random bit strings. The bilinaer GGM [14] extends this to provision a bilinear pairing e between groups, acting in a similar way. The only operation an adversary can perform locally is equality testing (by comparison of bitstrings), while to perform the group operation or bilinear pairing they must interact with their oracles. A result of using the GGM to model a group, it implicity assumes that the group operations themselves do not leak. Therefore, in practice, the group operations would have to be implemented using counter-measures.

We also assume the existence of a secure hash function, which we model in the random oracle model [9]. The random oracle is publicly accessible, and so we provide both the adversary and their leakage functions implicit access to it. We need not consider additional leakage from random oracle queries, since in every use case their inputs will already be known to the adversary or the leakage function.

Caveats and limitations. Our proofs also assume evaluation of an arbitrary sized input hash function takes a fixed amount of effort. If the hash did not support this functionality, the advantage of our two general constructions would be slightly different as the PRLF and MAC hash more data in SIVAT than in A5.

Another caveat to our results is that this particular instantiation provides indistinguishability from a stream of random elliptic curve group elements, and not indistinguishability from random bit strings. While this is a slightly weaker notion of security, it still implies the key requirements of traditional left-or-right, real-or-random security [6]. Work by a number of authors on "Elligators" [3,11,55] has investigated how to efficiently convert pseudo-random elliptic curve points into pseudo-random bitstrings, but are not without issue.

B.3 A Leakage-resiliant PRF

Since at the base level a PRLF instantiates the majority of our components for the generic composition, we begin by constructing this. There have been two leakage resilient PRFs in the continuous leakage model [16,18], but neither provides adaptive security. We claim that $\mathcal{T}$ is a secure PRF in the classical setting, and can be implemented as a PRLF. Thus we begin by showing that the output is indeed pseudo random in the leakage-free case. The proof is extremely similar to that for the original MAC, except the reduction goes to a decisional bilinear Diffie-Hellman assumption, rather than the computational version used in their work.

Theorem 7. *Let $\mathbb{A}$ be an adversary against the query PRF security of $\mathcal{T} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$. Then there exists an adversary $\mathbb{B}$ (of similar complexity to $\mathbb{A}$) who can break the DTBDH assumption for $\mathbb{BG}$, such that*

$$\mathbf{Adv}_{\mathcal{T}}^{\mathrm{PRF}}(\mathbb{A}) \le q_h \cdot q_r \cdot \mathbf{Adv}_{\mathbb{BG}}^{\mathrm{DTBDH}}(\mathbb{B}),$$

where q_h is the number of queries made to the random oracle and q_r is the number of queries to the challenge oracle.

```
**adversary** $\mathbb{A}^{O,\,\ell[\boldsymbol{\mathcal{T}}]_k}$
    $M, M' \leftarrow_{\$} \mathbb{G}_1$
    Query $Y \leftarrow O(M)$
    Set $L^{◓}(S_i^{◓}, M; r_{i+1}) := |e(S_i^{◓}, H(M))|_\lambda$
    Set $L^{◒}(S_i^{◒}, H(M), Y_i^{◓}, r_{i+1}^{◓}) = |Y \cdot e(S_i^{◒}, H(M))^{-1}|_\lambda$
    Query $Y', \Lambda^{◓}, \Lambda^{◒} \leftarrow \ell[\boldsymbol{\mathcal{T}}]_k(M', L^{◓}, L^{◒})$
    **return** $(\Lambda^{◓} = \Lambda^{◒})$
```

Fig. 11: An adversary $\mathbb{A}$ against the $\mathcal{L}_{\mathsf{T}_2}$–PRLF security of $\mathcal{T}$. With two generic group elements and two oracle queries the adversary distinguishes whether O implements $\mathcal{T}$ or is ideal with probability $1-2^{-\lambda}$.

```
**function** $\boldsymbol{\mathcal{T}}(M; r_{i+1}^{◓}, r_{i+1}^{\ominus})$
    $S_{i+1}^{◓}, W, Y_i^{◓} \leftarrow \mathcal{T}^{◓}(S_i^{◓}, M; r_{i+1}^{◓})$
    $S_{i+1}^{\ominus}, Y_i^{\ominus} \leftarrow \mathcal{T}^{\ominus}(S_i^{\ominus}, W; r_{i+1}^{\ominus})$
    $S_{i+1}^{◒}, Y_i \leftarrow$
      $\mathcal{T}^{◒}(S_i^{◒}, W, Y_i^{◓}, Y_i^{\ominus}; r_{i+1}^{◓}, r_{i+1}^{\ominus})$
    **return** $Y_i$

**function** $\mathcal{T}^{◓}(S_i^{◓}, M; r_{i+1}^{◓})$
    $W \leftarrow H(M)$
    $Y_i^{◓} \leftarrow e(S_i^{◓}, W)$
    $S_{i+1}^{◓} \leftarrow S_i^{◓} \cdot r_{i+1}^{◓}$
    **return** $(S_{i+1}^{◓}, W, Y_i^{◓})$
**function** $\mathcal{T}^{\ominus}(S_i^{\ominus}, W; r_{i+1}^{\ominus})$
    $Y_i^{\ominus} \leftarrow e(S_i^{\ominus}, W)$
    $S_{i+1}^{\ominus} \leftarrow S_i^{\ominus} \cdot r_{i+1}^{\ominus}$
    **return** $(S_{i+1}^{\ominus}, Y_i^{\ominus})$
**function** $\mathcal{T}^{◒}(S_i^{◒}, W, Y_i^{◓}, Y_i^{\ominus}; r_{i+1}^{◓}, r_{i+1}^{\ominus})$
    $Y_i^{◒} \leftarrow e(S_i^{◒}, W)$
    $Y_i \leftarrow Y_i^{◓} \cdot Y_i^{\ominus} \cdot Y_i^{◒}$
    $S_{i+1}^{◒} \leftarrow S_i^{◒} \cdot (r_{i+1}^{◓} \cdot r_{i+1}^{\ominus})^{-1}$
    **return** $(S_{i+1}^{◒}, Y_i)$
```

Fig. 12: A secure PRLF implementation $\boldsymbol{\mathcal{T}}$ using three shares.

Given the traditional similarities between a MAC and a PRF, it would be reasonable to expect that, given that $\mathcal{T}$ is both a PRF and $(\mathcal{L}_{\mathsf{T}_2}, \emptyset)$-LMAC secure, it might also be $\mathcal{L}_{\mathsf{T}_2}$-PRLF. However, this is not the case, once again highlighting the surprising effects of allowing adaptive leakage. Figure 11 defines an adversary that can distinguish between a real and random challenge (even in the GGM) when the leakage set is all leakage functions which output λ bits from each half of the computation (while obeying the OCLI assumption).

When the challenge is a real PRF call, $Y = e(k, H(M))$ and so $\Lambda_i^{◓}$ is λ bits of $e(S_i^{◓}, H(M))$ and $\Lambda_i^{◒}$ is λ bits of $Y \cdot e(S_i^{◒}, H(M))^{-1} = e(S_i^{◓}, H(M))$ and therefore the two leakage results are equal with probability one. However, when Y is from \$, $\Lambda_i^{◒}$ will provide λ bits of a random element and therefore the two outputs are the same with probability $2^{-\lambda}$. Hence the adversary can win the PRLF game with probability $1 - 2^{-\lambda}$.

Constructing a secure PRLF. While it is not possible to show PRLF security for an implementation with two sections, we are able to recover security with just one further share, forming the three share implementation shown in Figure 12. The proof reduces from the leakage setting to the leak-free case, which we already proved secure. Following the OCLI assumption, it is proven secure for leakage set $\mathcal{L}_{\mathsf{T}}$ containing all functions of the form:

$$\begin{aligned}&L_T((S^{◓}, S^{\ominus}, S^{◒}), M; (r^{◓}, r^{\ominus}))\\&\quad = (L^{◓}(S^{◓}, M; r^{◓}), L^{\ominus}(S^{\ominus}, H(M); r^{\ominus}), L^{◒}(S^{◒}, H(M), Y^{◓}, Y^{\ominus}; r^{◓}, r^{\ominus})),\end{aligned}$$

```
function $\boldsymbol{\mathcal{V}}_k(M, T; r^{\text{◓}}_{i+1}, r^{\ominus}_{i+1})$
    $S^{\text{◓}}_{i+1}, W, T^{\text{◓}}_i \leftarrow \mathcal{V}^{\text{◓}}(S^{\text{◓}}_i, M; r^{\text{◓}}_{i+1})$
    $S^{\ominus}_{i+1}, h^{\ominus}_i \leftarrow \mathcal{V}^{\ominus}(S^{\ominus}_i, W, T^{\text{◓}}_i; r^{\ominus}_{i+1})$
    $S^{\text{◒}}_{i+1}, b_i \leftarrow$
        $\mathcal{V}^{\text{◒}}(S^{\text{◒}}_i, W, h^{\ominus}_i, T; r^{\text{◓}}_{i+1}, r^{\ominus}_{i+1})$
    return $b_i$

function $\mathcal{V}^{\text{◓}}(S^{\text{◓}}_i, M; r^{\text{◓}}_{i+1})$
    $W \leftarrow H(M)$
    $T^{\text{◓}}_i \leftarrow e(S^{\text{◓}}_i, W)$
    $S^{\text{◓}}_{i+1} \leftarrow S^{\text{◓}}_i \cdot r^{\text{◓}}_{i+1}$
    return $(S^{\text{◓}}_{i+1}, W, T^{\text{◓}}_i)$
function $\mathcal{V}^{\ominus}(S^{\ominus}_i, W, T^{\text{◓}}_i; r^{\ominus}_{i+1})$
    $T^{\ominus}_i \leftarrow T^{\text{◓}}_i \cdot e(S^{\ominus}_i, W)$
    $h^{\ominus}_i \leftarrow H'(T^{\ominus}_i)$
    $S^{\ominus}_{i+1} \leftarrow S^{\ominus}_i \cdot r^{\ominus}_{i+1}$
    return $(S^{\ominus}_{i+1}, h^{\ominus}_i)$
function $\mathcal{V}^{\text{◒}}(S^{\text{◒}}_i, W, h^{\ominus}_i, T; r^{\text{◓}}_{i+1}, r^{\ominus}_{i+1})$
    $T^{\text{◒}}_i \leftarrow T \cdot e(S^{\text{◒}}_i, W)^{-1}$
    $h^{\text{◒}}_i \leftarrow H'(T^{\text{◒}}_i)$
    $b_i \leftarrow (h^{\ominus}_i = h^{\text{◒}}_i)$
    $S^{\text{◒}}_{i+1} \leftarrow S^{\text{◒}}_i \cdot (r^{\text{◓}}_{i+1} \cdot r^{\ominus}_{i+1})^{-1}$
    return $(S^{\text{◒}}_{i+1}, b_i)$
```

Fig. 13: A secure implementation $\boldsymbol{\mathcal{V}}$. The two-share version $\boldsymbol{\mathcal{V}}_2$ merges $\mathcal{V}^{\text{◓}}$ and $\mathcal{V}^{\ominus}$.

where $L^{\text{◓}}$, $L^{\ominus}$ and $L^{\text{◒}}$ output λ bits, $S^{\text{◓}}, S^{\ominus}$ and $S^{\text{◒}}$ is the secret information used by each part, $r^{\text{◓}}, r^{\ominus}$ is the randomness, and $Y^{\text{◓}}, Y^{\ominus}$ is the information passed from the first two parts of the function to the final part.

Theorem 8. *Let $\mathbb{A}$ be an adversary against the $\mathcal{L}_{\mathrm{T}}$-PRLF security of $\boldsymbol{\mathcal{T}}$ in the generic group model, then there exists an adversary $\mathbb{A}_{\mathrm{PRF}}$ (of similar complexity to $\mathbb{A}$) against the PRF security of $\mathcal{T}$ such that*

$$\mathbf{Adv}^{\mathrm{PRLF}}_{\boldsymbol{\mathcal{T}};\mathcal{L}_{\mathrm{T}}}(\mathbb{A}) \leq 2^{4\cdot\lambda} \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathcal{T}}(\mathbb{A}_{\mathrm{PRF}}) + \frac{\gamma^2}{p},$$

where γ is the number generic group elements, p is the size of the group and λ is the amount of leakage allowed per share of the PRF, giving $3 \cdot \lambda$ bits leakage per function call.

B.4 A fully Leakage-resilient MAC

The MOSW verification routine uses the classical method of recomputing a tag to verify correctness. However, this is not secure under verification leakage, since an attacker may leak (functions of) the candidate tag. Thus providing a MAC secure under verification leakage was left as an open problem.

In this work we answer this, providing a secure verification implementation $\boldsymbol{\mathcal{V}}$ in Figure 13. Instead of calculating a candidate tag, we invert the final pairing step, and surrounding the comparison in a random oracle query, meaning the candidate tag is never available to the adversary.

For simplicity, consider the (also secure) two share version $\boldsymbol{\mathcal{V}}_2$. If T is a valid tag on message M under key $k = S^{\text{◓}} \cdot S^{\text{◒}}$, then $T = e(k, H(M))$. Instead of calculating $T' = e(S^{\text{◓}}, H(M)) \cdot e(S^{\text{◒}}, H(M))$ and testing whether $T' = T$, we calculate the two pairings and check whether $e(S^{\text{◓}}, H(M)) = T \cdot e(S^{\text{◒}}, H(M))^{-1}$. However, rather than doing this final comparison directly, we enclose each side in a random oracle call to ensure the adversary can never leak on both parts of the tag in unison. This results in the comparision of $H'(e(S^{\text{◓}}, H(M))) = H'(T \cdot e(S^{\text{◒}}, H(M))^{-1})$ This allows us to recover security even under a large set of verification leakage functions, meaning both $(\boldsymbol{\mathcal{T}}_2, \boldsymbol{\mathcal{V}}_2)$ and $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ define secure LMAC implementations under the OCLI assumption.

For our overall implementation, we require that the tagging algorithm is a PRLF, and so use the three share variant $\boldsymbol{\mathcal{V}}$, since key updating agrees with $\boldsymbol{\mathcal{T}}$. To prove security under OCLI, we define the

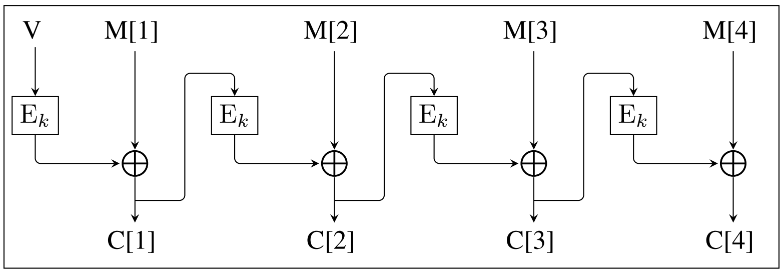

Fig. 14: CFB Mode of Operation. The message M is parsed into blocks $M[1]||\ldots||M[m]$, and fed through with the routine to output ciphertext $C[1]||\ldots||C[m]$. It is secure if the initial value V is random and E_k is a PRF.

verification leakage set $\mathcal{L}_\mathsf{V}$ to be the set of functions of the form:

$$\begin{aligned} &L_V((S^{◓}, S^{◒}), M, T; (r^{◓}, r^{\ominus})) \\ &\quad = (L_V^{◓}(S^{◓}, M; r^{◓}), L_V^{\ominus}(S^{\ominus}, H(M), T^{◓}; r^{\ominus}), L_V^{◒}(S^{◒}, H(M), T, T^{\ominus}; r^{◓}, r^{\ominus})) \end{aligned}$$

where $L^{◓}$, $L^{\ominus}$ and $L^{◒}$ each output λ bits, $S^{◓}$, $S^{\ominus}$ and $S^{◒}$ is the secret information used by each share, $r^{◓}$, $r^{\ominus}$ the randomness, and $T_i^{◓}$, $T_i^{\ominus}$ the information passed between the shares. This allows us to state the following theorem, reducing to the security in the leakage-free case, shown secure by MOSW.

Theorem 9. *Let $\mathbb{A}$ be an adversary against the $(\mathcal{L}_\mathsf{T}, \mathcal{L}_\mathsf{V})$-sEUF–CMLA security of $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}})$ in the generic group model, for H' a random permuation. Then, there exists an adversary $\mathbb{A}_{\mathrm{MAC}}$ (of similar complexity to $\mathbb{A}$) against the sEUF–CMLA security of $(\mathcal{T}, \mathcal{V})$ such that*

$$\mathbf{Adv}^{\mathrm{sEUF-CMLA}}_{\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{V}};\mathcal{L}_\mathsf{T},\mathcal{L}_\mathsf{V}}(\mathbb{A}) \leq 2^{4\cdot\lambda} \cdot \mathbf{Adv}^{\mathrm{sEUF-CMA}}_{\mathcal{T},\mathcal{V}}(\mathbb{A}_{\mathrm{MAC}}) + \frac{\gamma^2}{p},$$

where γ is the total number of group elements, p is the size of the group and λ is the amount of leakage output by each of the three parts of the leakage function.

B.5 A Leakage-resiliant IV-based Encryption Scheme

The final component required is an IND–aCPLA secure encryption $\mathbf{iv}\boldsymbol{\mathcal{E}}$. We find that CFB mode (Figure 14) is secure against leakage when instantiated with a PRLF, using a proof that follows the original proof for CFB without leakage [1]. To instantiate it, we use $\boldsymbol{\mathcal{T}}$, since it is a PRLF secure against leakage set $\mathcal{L}_\mathsf{T}$. Then, we define the leakage set $\mathcal{L}_\mathsf{ivE}$ to be the collection of all functions $L_{CFB} : \mathsf{K}\times\mathsf{N}\times\mathsf{M}\times\mathsf{R} \to \{0,1\}^*$ that are of the form

$$L_{CFB}(K, V, M; R) = \{L_i(K, C_i; R_i)\}_{i=0}^{n-1}$$

where for $i > 0$ C_i is the i^{th} block of ciphertext $C = iv\mathcal{E}_k(S, V, M)$ and $C_0 = V$, M is an n block message, R_i is the randomness passed to the i^{th} PRLF call and $L_i \in \mathcal{L}_\mathsf{T}$ is some leakage function of the PRLF.

This corresponds to applying the OCLI assumption to the design of CFB encryption. It is easy to see from the diagram above that this includes every internal computation, but does not allow the adversary to compute functions of variables not used together. Since the number of components of a leakage function is only bounded by the length of the associated message, leakage means that the longer message, the more leakage the adversary receives, modelling the fact the longer a device runs the more leakage might can be captured from it (*e.g.* a longer power trace).

Being a stream cipher, CFB mode encryption and decryption are extremely similar. Encryption takes the previous ciphertext block (starting with the IV), runs it through the PRF and xors it to the plaintext block to produce the next ciphertext block. Decryption takes in the previous ciphertext block (starting with the IV), runs it through the PRF and xors it to the current ciphertext block to produce the plaintext block. Due to the similarities, it makes sense that both encryption and decryption leak the same. Thus we fix the decryption leakage set to be the same as the encryption leakage, $\mathcal{L}_{\mathsf{ivD}} = \mathcal{L}_{\mathsf{ivE}}$.

Theorem 10. *Let* $(\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}})$ *be the symmetric encryption scheme formed from running the PRLF* $\boldsymbol{\mathcal{T}}$ *in CFB mode. Let* $\mathbb{A}$ *be an adversary against the* $(\mathcal{L}_{\mathsf{ivE}}, \mathcal{L}_{\mathsf{ivD}})$*-IND–aCPLA security of* $(\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}})$*. Then there exists an adversary* $\mathbb{B}$ *(of similar complexity to* $\mathbb{A}$*) against the* $\mathcal{L}_{\mathsf{T}}$*-PRLF security of* $\boldsymbol{\mathcal{T}}$ *such that*

$$\mathbf{Adv}^{\mathrm{IND-aCPLA}}_{\mathbf{iv}\boldsymbol{\mathcal{E}},\mathbf{iv}\boldsymbol{\mathcal{D}};\mathcal{L}_{\mathsf{ivE}},\mathcal{L}_{\mathsf{ivD}}}(\mathbb{A}) \leq \mathbf{Adv}^{\mathrm{PRLF}}_{\boldsymbol{\mathcal{T}};\mathcal{L}_{\mathsf{T}}}(\mathbb{A}_{\mathrm{PRF}}) + \frac{q_e^2 \cdot s^2}{|\mathsf{T}|},$$

where σ *is the total number of blocks encrypted and* $|\mathsf{T}|$ *is the blocksize.*

B.6 nLAE and mrLAE constructions in the generic group model

Collecting together the bounds for each component with the security of the $\mathsf{A5}$ and SIVAT composition mechanisms, we are able to give the first provably secure leakage resilient AE scheme. Carefully counting the number of generic group queries γ made by the different elements of the construction, we find that $\gamma \leq 26q + 14\sigma + 3g$, where g is the total number of direct queries the adversary makes to the generic group or random oracle, q the number of AE oracle queries (honest or challenge) of total σ blocks.

Theorem 11. *Let* $\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}$ *be the primitives defined in this section, and* $\mathsf{A5} = \mathsf{A5}[\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{T}}; \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}]$*. Then for any adversary* $\mathbb{A}$*,*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{LAE}}_{\mathsf{A5};\mathcal{L}_{\mathsf{Enc}},\mathcal{L}_{\mathsf{Dec}}}(\mathbb{A}) &\leq \frac{2^{4\lambda} \cdot (g \cdot (2q+\sigma) + 3) \cdot \gamma^2 + \sigma^2}{p} \\ &\leq \frac{27 \cdot 2^{4\lambda} \cdot (g \cdot \sigma + 1) \cdot (9q + 5\sigma + g)^2}{p}.\end{aligned}$$

Theorem 12. *Similarly,* $\mathsf{SIVAT}[\mathbf{iv}\boldsymbol{\mathcal{E}}, \mathbf{iv}\boldsymbol{\mathcal{D}}; \boldsymbol{\mathcal{T}}; \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{V}}]$ *is mrLAE secure, with the same bound as Theorem 11.*

If Shrimpton and Terashima's [53] (weaker) notion were considered to be sufficient, where "recovery information" is not required to be random and hence the LMAC's tagging algorithm need not also be a LPRF, we would get even better bounds.

C Security Proofs for Leakage Resilient Components

In this section we provide the proofs (and any supporting material) for all the theorems given in Section B.

Before we do so, we first introduce a bilinear Diffie-Hellman problem.. The problem introduced is the decisional variant of the problem used to prove the security of the MAC by Martin *et al.* [34].

Definition 13 (Decisional Target Bilinear Diffie–Hellman Problem (DTBDH)). *Let* $\mathbb{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, p)$ *be a set of groups with a pairing between them. The decisional target bilinear Diffie-Hellman problem is then defined as; given* $g_1, g_2, g_2^x, g_3^y, g_3^z$ *determine if* $g_3^z = g_3^{x \cdot y}$ *or* $g_3^z = g_3^r$*, where* x, y, r *are chosen uniformly at random from* $\mathbb{Z}_p$ *and* $z = x \cdot y$ *if* $b = 1$ *and* r *otherwise for* b *chosen uniformly at random from* $\{0, 1\}$*. Given an adversary* $\mathbb{A}$*, their advantage is defined as:*

$$\mathbf{Adv}^{\mathrm{DTBDH}}_{\mathbb{BG}}(\mathbb{A}) := \Pr[\mathbb{A}(g_1, g_2, g_2^x, g_3^y, g_3^{xy}) = 1] - \Pr[\mathbb{A}(g_1, g_2, g_2^x, g_3^y, g_3^r) = 1].$$

Now that the definition is in place we are able to give the proof of the theorem statements.

```
adversary $\mathbb{B}(g_1, g_2, g_2^x, g_3^y, g_3^z)$
    $j \leftarrow 0$
    $m \leftarrow 0$
    $s \leftarrow\$ [q_h]$
    $b' \leftarrow \mathbb{A}^{H(\cdot),\mathrm{RR}(\cdot,\cdot)}()$
    return $b'$

simulator $H(X)$
    $m \leftarrow m + 1$
    if $m == s$ then
        $W[X] \leftarrow \times$
        return $g_2^x$
    else
        if $W[X] == \perp$ then
            $W[X] \leftarrow\$ \mathbb{Z}_p$
        return $g_2^{W[X]}$
```

```
simulator $\mathrm{RR}(X)$
    $j \leftarrow j + 1$
    if $j > i$ then
        $Y \leftarrow\$ \mathbb{G}_3$
    else if $j == i$ then
        if $W[X] == \times$ then
            $Y \leftarrow g_3^z$
        else
            ABORT
    else
        if $W[X] == \times$ then
            ABORT
        if $W[X] == \perp$ then
            $W[X] \leftarrow\$ \mathbb{Z}_p$
        $Y \leftarrow (g_3^y)^{W[X]}$
    return $Y$
```

Fig. 15: Adversary $\mathbb{B}$ simulates the hybrid games H_{i-1}, H_i for adversary $\mathbb{A}$

Proof (Of Theorem 7). Let $\mathbb{A}$ be an adversary against the query PRF security of $\mathcal{T}$ (assume without loss of generality the adversary makes unique queries), then it is possible to construct a series of hybrid games (given in Fig. 15). In game H_i the first i queries correspond to real queries, while the remaining queries are responded to with random queries. Game H_{q_r} corresponds to the real world while H_0 corresponds to the random world. If $\mathbb{A}$ can distinguish between these two games with probability ϵ there exists two consecutive games H_{i-1}, H_i that they can distinguish between, with probability at least $\frac{\epsilon}{q_r}$. Figure 15 shows how an adversary $\mathbb{B}$ against the DTBDH assumption can simulate these two games for adversary $\mathbb{A}$.

Assume without loss of generality that the value X sent to the PRF on the i^{th} query was sent to the random oracle beforehand. If $\mathbb{B}$ can guess which query to the random oracle the value X was sent, the reduction will behave as expected. As the oracle will return g_3^z, which is either g_3^{xy} or g_3^r corresponding to the two hybrids H_i or H_{i-1}. The probability that this happens is $\frac{1}{q_h}$.

When the adversary $\mathbb{B}$ can guess which oracle query the i^{th} challenge query is, all queries $j > i$ return random results, while if $j < i$ the RR oracle returns real values (it will not trigger the abort since the X which has the flag set is query i). Query i in this scenario returns the DTBDH challenge. It then follows that z is real or random depending if the adversary is playing hybrid H_i or H_{i-1} respectively. Thus, if adversary $\mathbb{A}$ can distinguish between these two hybrid games, $\mathbb{B}$ can win the DTBDH game. Putting it all together gives the desired result. $\square$

Proof (Of Theorem 8). The proof is given in the Generic Group Model and shows that the use of leakage does not allow the adversary to learn any elements that they would be unable to learn if no leakage had been involved. Once this has been shown, it follows that the adversary's advantage can at most be increased by the number of bits that can be learnt about a single element. By showing that each element is only leaked, at most, four times, the adversary's advantage can, at most, be increased by $2^{4\cdot\lambda}$ over the advantage in the game where no leakage is involved.

Group elements will be represented by polynomials, which will be instantiated at the end of the computation. The polynomials allow the game to keep track of which elements the adversary has asked for in a straightforward manner and because they are instantiated at the end of the computation, the adversary's decisions clearly cannot depend on the actual values of the elements. It must be shown that the chance of these polynomials colliding when evaluated is small (it will be shown to be $\frac{q^2}{p}$). Since the PRLF involves three groups, polynomials will be tracked per group.

Let $\mathcal{K}$, $\{\mathcal{R}_j^{◓}\}_{j=0}^{q_R}\{\mathcal{R}_j^{\ominus}\}_{j=0}^{q_R}$, $\{\mathcal{H}_i\}_{j=1}^{q_H}$, $\{\mathcal{U}_j\}_{j=1}^{2\cdot q_O}$, $\{\mathcal{V}_j\}_{j=1}^{2\cdot q_O}$, $\{\mathcal{W}_j\}_{j=1}^{2\cdot q_O}$, $\{\mathcal{Y}_j\}_{j=0}^{q_R}$ be indeterminants where q_H is the number of hash queries, q_R is the number of calls to the challenge oracle, q_F is the number of calls made to the PRF oracle and q_O is the number of group oracle calls (by the adversary). The indeterminants represent the following; $\mathcal{K}$ is the secret key, $\{\mathcal{R}_j^{\text{◓}}\}_{j=0}^{q_R}$, $\{\mathcal{R}_j^{\ominus}\}_{j=0}^{q_R}$ are the randomness used to update the key, $\{\mathcal{Y}_j\}_{j=0}^{q_R}$ is the output from the challenge oracle, $\{\mathcal{H}_i\}_{j=1}^{q_H}$ represent any hash function queries and $\{\mathcal{U}_j\}_{j=1}^{2\cdot q_O}, \{\mathcal{V}_j\}_{j=1}^{2\cdot q_O}, \{\mathcal{W}_j\}_{j=1}^{2\cdot q_O}$ represent any elements that are guessed in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ respectively. Let $q = q_H + 8 \cdot q_R + 8 \cdot q_F + 3 \cdot q_O$, the factor 3 arises from the fact the adversary can guess the representation of two elements being passed into a binary operation and learns another from the output, thus adding, at most, 3 elements to the list per oracle call. Calling either the challenge or PRF oracle adds 8 polynomials to the list; the 3 updated shares, the PRF output and an extra element (since the calculuation requires two multiplications) and the three intermediate values $Y^{\text{◓}}, Y^{\ominus}, Y^{\text{◒}}$. The lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ are used to keep track of polynomials and their representations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ respectively. They are initialised as follows:

$$\begin{aligned}
\mathcal{L}_1 &= \{(1, \xi_1^1)\} \cup \{(\mathcal{R}_j^{\text{◓}}, \xi_{i+2}^1)\}_{j=0}^{q_R} \cup \{(\mathcal{R}_j^{\ominus}, \xi_{i+q_R+2}^1)\}_{j=0}^{q_T} \\
\mathcal{L}_2 &= \{(1, \xi_1^2)\} \\
\mathcal{L}_3 &= \{(1, \xi_1^3)\}
\end{aligned}$$

where the ξ_j^i are chosen uniformly at random from Ξ^i, such that all polynomials have a unique representation. The sets the representations are drawn from Ξ^1, Ξ^2, Ξ^3 (for $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ respectively) are all disjoint. All three lists are initially instantiated with the identity. Note that it is not strictly necessary to instantiate the identity in $\mathbb{G}_3$ since it can be calculated using the other information provided. We precompute the representations of the randomness used for the key update, the $r_i^{\text{◓}}$'s and $r_i^{\ominus}$'s, but since the adversary does not have access to this list of elements, this does not effect the game.

The Adversary $\mathbb{A}$ outputs a bit b' and is said to have won if:

1. $\mathcal{F}_i^l = \mathcal{F}_j^l$ for $l \in \{1, 2, 3\}$ and $i \neq j$
2. $b' = b$

The first case corresponds to the adversary being able to create two polynomials which evaluate to the same value. If this occurs then a single group element has two representations. The adversary is said to have won because the simulation has been broken. The second case corresponds to the adversary being able to distinguish which world they are in. The only way (beyond guessing) that an adversary can distinguish between the two worlds is if they can construct a polynomial $\mathcal{F}_i^3$ such that $\mathcal{F}_i^3 - \mathcal{K} \cdot \mathcal{H} = 0$. Where $\mathcal{H}$ is the indeterminant corresponding to the hash of an X that has not been sent to the PRF oracle. This corresponds to the adversary winning the PRLF game. We first bound the chance of a collision and then go on to bound the chance of winning the game.

All polynomials originally in $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ are of degree one and the only operation that increases the degree is the pairing operation which can only be called on elements in $\mathbb{G}_1, \mathbb{G}_2$. This means that degree two polynomials can be in $\mathbb{G}_3$ but not the other two lists (since there is no way to get an element of $\mathbb{G}_3$ into either of the other two groups). Hence, by the Schwartz-Zippel lemma, the probability of two (non-zero) polynomials evaluating to the same value is $\frac{2}{p}$. Since there are, at most, q polynomials, there are $\binom{q}{2} \leq \frac{q^2}{2}$ pairs of polynomials that could collide and thus the probability of any two polynomials colliding is $\frac{q^2}{p}$.

Without loss of generality, we will now only look at leakage in the target group $\mathbb{G}_3$ since any element from $\mathbb{G}_1, \mathbb{G}_2$ calculated by the leakage can be transferred over to $\mathbb{G}_3$ using a pairing, with the corresponding generator, and any elements known to the adversary can easily be embedded into the leakage by the adversary as required. Since, any element which can be leaked upon from $\mathcal{L}_1$ or $\mathcal{L}_2$ can be leaked on from $\mathcal{L}_3$, by transferring the element to $\mathbb{G}_3$ with a pairing, this does not weaken the adversary.

While only the leakage from $\mathcal{L}_3$ needs to be considered, due to the OCLI assumption $F^{\text{◓}}$, $F^{\ominus}$ and $F^{\text{◒}}$ will have access to different secret information and, due to the randomness used to update the key, each iteration will have access to different secret information. The lists $\mathcal{L}_i^{◓}$, $\mathcal{L}_i^{\ominus}$ and $\mathcal{L}_i^{◒}$ will represent all elements which can be calculated by $L_i^{◓}$, $L_i^{\ominus}$ and $L_i^{◒}$ respectively.

Let $\mathcal{L}_i^{◓}$ be the set of elements that could be computed by the leakage function $L_i^{◓}$. Then, utilising the leakage functions from the theorem statement:

$$\mathcal{L}_i^{◓} = \{A \cdot \mathcal{S}_i^{◓} + B \cdot \mathcal{R}_{i+1}^{◓} + C\}$$

Where $A, B \in \mathcal{F}_p[\{\mathcal{H}_j\}_{j=1}^{q_H}, \{\mathcal{V}_j\}_{j=1}^{2q_O}]$ and $C \in \mathcal{F}_p[\mathcal{K}\{\mathcal{H}_j\}_{j=1}^{i-1}, \{\mathcal{H}_j\}_{j=1}^{q_H}, \{\mathcal{Y}_j\}_{j=1}^{q_R}, \{\mathcal{U}_j\}_{j=1}^{2q_O}, \{\mathcal{V}_i\}_{i=1}^{2q_O}, \{\mathcal{W}_i\}_{i=1}^{q_O}]$ and $\mathcal{S}_i^{◓}$ denotes $\sum_{j=0}^{i} \mathcal{R}_j^{◓}$ (corresponding to the definition of $S_i^{◓}$ in the PRLF).

Let $\mathcal{L}_i^{\ominus}$ be the set of elements that could be computed by the leakage function $L_i^{\ominus}$. Therefore:

$$\mathcal{L}_i^{\ominus} = \{A \cdot \mathcal{S}_i^{\ominus} + B \cdot \mathcal{R}_{i+1}^{\ominus} + C\}$$

Where $\mathcal{S}_i^{\ominus}$ denotes $\sum_{j=0}^{i} \mathcal{R}_j^{\ominus}$ (corresponding to the definition of $S_i^{\ominus}$ in the PRLF).

Let $\mathcal{L}_i^{◒}$ be the set of elements that could be computed by the leakage function $L_i^{◒}$. Therefore:

$$\mathcal{L}_i^{◒} = \{A \cdot \mathcal{S}_i^{◒} + B^{◓} \cdot \mathcal{R}_{i+1}^{◓} + B^{\ominus} \cdot \mathcal{R}_{i+1}^{\ominus} + C + d^{◓} \cdot \mathcal{S}_i^{◓} \cdot \mathcal{H}_i + d^{◓} \cdot \mathcal{S}_i^{\ominus} \cdot \mathcal{H}_i\}$$

Where $d^{◓}, d^{\ominus} \in \mathcal{F}_p$, $B^{◓}, B^{\ominus} \in \mathcal{F}_p[\{\mathcal{H}_j\}_{j=1}^{q_H}, \{\mathcal{V}_j\}_{j=1}^{2q_O}]$ and $\mathcal{S}_i^{◒}$ denotes $\mathcal{K} - \sum_{j=0}^{i}(\mathcal{R}_j^{◓} + \mathcal{R}_j^{\ominus})$ (corresponding to the definition of $S_i^{◒}$ in the PRLF). Without loss of generality we will assume that ith PRF call maps to $\mathcal{H}_i$.

The adversary can win if they can leak $\mathcal{H} \cdot \mathcal{K}$ where $\mathcal{H}$ corresponds to some unqueried value X. However, there is no such linear combination within the leakage sets that allows this to be possible.

To bound the leakage per element, we will only consider leakage functions which contains at least one unknown group element. Since, while completely known elements can be leaked on multiple times, they do not give the adversary any new information. By showing that each element can only leak a bounded number of times (4 times), the adversary cannot learn any group elements which they would not be able to learn when leakage is not involved and thus the advantage will be increased by at most the number of bits the adversary can learn.

- $\mathcal{S}_i^{◓}$ can be leaked on twice; once in $\mathcal{L}_{i-1}^{◓}$ since $S_{i-1}^{◓}$ is passed in and $r_i^{◓}$ is generated internally (represented by the polynomials $\mathcal{S}_{i-1}^{◓}$ and $\mathcal{R}_i^{◓}$), and once in $\mathcal{L}_i^{◓}$ since it is passed in.
- $\mathcal{S}_i^{\ominus}$ can be leaked on twice; once in $\mathcal{L}_{i-1}^{\ominus}$ since $S_{i-1}^{\ominus}$ is passed in and $r_i^{\ominus}$ is generated internally (represented by the polynomials $\mathcal{S}_{i-1}^{\ominus}$ and $\mathcal{R}_i^{\ominus}$), and once in $\mathcal{L}_i^{\ominus}$ since it is passed in.
- $\mathcal{S}_i^{◒}$ can be leaked on twice; once each in $\mathcal{L}_{i-1}^{◒}$ and $\mathcal{L}_i^{◒}$ due to a similar argument as above.
- $\mathcal{R}_{i+1}^{◓}$ can be leaked on twice; once each in $\mathcal{L}_i^{◓}$ and $\mathcal{L}_i^{◒}$ since it is generated in $F^{◓}$ and then passed into $F^{◒}$ on the i^{th} iteration.
- $\mathcal{R}_{i+1}^{\ominus}$ can be leaked on twice; once each in $\mathcal{L}_i^{\ominus}$ and $\mathcal{L}_i^{◒}$ since it is generated in $F^{\ominus}$ and then passed into $F^{◒}$ on the i^{th} iteration.
- $\mathcal{S}_i^{◓} \cdot \mathcal{H}_i$, the intermediated state, can be leaked on 4 times, once in each of; $\mathcal{L}_{i-1}^{◓}$, $\mathcal{L}_i^{◓}$, $\mathcal{L}_{i-1}^{◒}$, $\mathcal{L}_i^{◒}$ using the argument above for calculating the next share and the fact for input X_i, on the i^{th} tag call the intermediate state is generated in $F^{◓}$ and passed into $F^{◒}$. To leak on this four times will require querying the same input twice.
- $\mathcal{S}_i^{\ominus} \cdot \mathcal{H}_i$, the intermediated state, can be leaked on 4 times, once in each of; $\mathcal{L}_{i-1}^{\ominus}$, $\mathcal{L}_i^{\ominus}$, $\mathcal{L}_{i-1}^{◒}$, $\mathcal{L}_i^{◒}$ using the argument above for calculating the next share and the fact for input X_i, on the i^{th} tag call the intermediate state is generated in $F^{\ominus}$ and passed into $F^{◒}$. To leak on this four times will require querying the same input twice.

Since each element can only be leaked on, at most, four times, the adversary can only learn up to $4 \cdot \lambda$ bits of information per unknown group element. Therefore, the adversary's advantage can be at most $2^{4 \cdot \lambda}$ times the advantage of playing the standard non-leakage game. This results in the bound given in the theorem statement. □

Proof (Of Theorem 9). In this proof we use the same initial setup of representing group elements as the PRLF proof, with the same indeterminants. In this game the adversary $\mathbb{A}$ outputs the pair (M, T) and is said to have won if:

1. $\mathcal{F}_i^l = \mathcal{F}_j^l$ for $l \in \{1, 2, 3\}$ and $i \neq j$
2. $\mathcal{K} \cdot \mathcal{H}^* - \mathcal{T} = 0$ where $\mathcal{H}^*$ is the indeterminant corresponding to the hash of M, $\mathcal{T}$ is the corresponding polynomial for T and T was not output from the tag oracle

The first case corresponds to the adversary being able to create two polynomials which evaluate to the same value. Two distinct polynomials evaluating to the same value, means that a single group element has two distinct representations. This breaks the simulation and therefore the adversary is said to have won the game. As previously this can be bounded as $\frac{q^2}{p}$ (for $q = q_H + 12 \cdot q_R + 8 \cdot q_T + 12 \cdot q_V + 3 \cdot q_O$). The second case corresponds to the adversary being able to create a forgery on the MAC. The polynomial given in the second case evaluating to zero implies that the adversary output a valid forgery for the MAC. Therefore, they have won the sEUF–CMLA game.

As before we consider what the i^{th} leakage function can leak on. We now have to consider leakage sets for both tag and verify queries. The tagging query leakage is the same as the leakage for the PRLF above but is recapped below.

$$\mathcal{L}_i^{\text{◓}} = \{A \cdot \mathcal{S}_i^{\text{◓}} + B \cdot \mathcal{R}_{i+1}^{\text{◓}} + C\}$$

$$\mathcal{L}_i^{\ominus} = \{A \cdot \mathcal{S}_i^{\ominus} + B \cdot \mathcal{R}_{i+1}^{\ominus} + C\}$$

$$\mathcal{L}_i^{\text{◒}} = \{A \cdot \mathcal{S}_i^{\text{◒}} + \mathcal{B}^{\text{◓}} \cdot \mathcal{R}_{i+1}^{\text{◓}} + \mathcal{B}^{\ominus} \cdot \mathcal{R}_{i+1}^{\ominus} + C + d^{\text{◓}} \cdot \mathcal{S}_i^{\text{◓}} \cdot \mathcal{H}_i + d^{\ominus} \cdot \mathcal{S}_i^{\ominus} \cdot \mathcal{H}_i\}$$

We then give the leakage sets for verify. It is important to note the extra element in the $\mathcal{J}_i^{\text{◒}}$ set (which was added on in the $\mathcal{L}_i^{\text{◒}}$ set). This cannot be added on in this set because it was passed through the second hash function H' first and therefore is no longer in the group.

$$\mathcal{J}_i^{\text{◓}} = \{A \cdot \mathcal{S}_i^{\text{◓}} + B \cdot \mathcal{R}_{i+1}^{\text{◓}} + C\}$$

$$\mathcal{J}_i^{\ominus} = \{A \cdot \mathcal{S}_i^{\ominus} + B \cdot \mathcal{R}_{i+1}^{\ominus} + C + d \cdot \mathcal{S}_i^{\text{◓}} \cdot \mathcal{H}_i\}$$

$$\mathcal{J}_i^{\text{◒}} = \{A \cdot \mathcal{S}_i^{\text{◒}} + \mathcal{B}^{\text{◓}} \cdot \mathcal{R}_{i+1}^{\text{◓}} + \mathcal{B}^{\ominus} \cdot \mathcal{R}_{i+1}^{\ominus} + C, d^{\text{◓}} \cdot \mathcal{S}_i^{\text{◓}} \cdot \mathcal{H}_i + d^{\ominus} \cdot \mathcal{S}_i^{\ominus} \cdot \mathcal{H}_i\}$$

The adversary can win if they can leak $\mathcal{H} \cdot \mathcal{K}$ where $\mathcal{H}$ corresponds to some unqueried value X. However, there is no such linear combination within the leakage sets that allows this to be possible.

To bound the leakage per element, we will only consider leakage functions which contains at least one unknown group element. Since, while completely known elements can be leaked on multiple times, they do not give the adversary any new information. By showing that each element can only leak a bounded number of times, the adversary cannot learn any group elements which they would not be able to learn when leakage is not involved and thus the advantage will be increased by at most the number of bits the adversary can learn.

A similar approach, to the one given for the PRLF, can be used to show that each (unknown) element can be leakage on at most four times. Since each element can only be leaked on, at most, four times, the adversary can only learn up to $4 \cdot \lambda$ bits of information per unknown group element. Therefore, the adversary's advantage can be at most $2^{4 \cdot \lambda}$ times the advantage of playing the standard non-leakage game. This results in the bound given in the theorem statement. □

```
function $O_1(M)$
    $M_1, \cdots, M_n \leftarrow M$
    $C_0 \leftarrow\!\$\ \mathsf{T}$
    for $i = 1$ to $n$ do
        $F_i \leftarrow \mathcal{F}_k(C_{i-1})$
        $C_i \leftarrow F_i \star M_i$
    $C \leftarrow C_0, \ldots, C_n$
    return $C$

function $O_2(M)$
    $M_1, \cdots, M_n \leftarrow M$
    $C_0 \leftarrow\!\$\ \mathsf{T}$
    for $i = 1$ to $n$ do
        $F_i \leftarrow \$(C_{i-1})$
        $C_i \leftarrow F_i \star M_i$
    $C \leftarrow C_0, \ldots, C_n$
    return $C$

function $O_3(M)$
    $M_1, \cdots, M_n \leftarrow M$
    $C_0 \leftarrow\!\$\ \mathsf{T}$
    for $i = 1$ to $n$ do
        $F_i \leftarrow\!\$\ \mathsf{T}$
        $C_i \leftarrow F_i \star M_i$
    $C \leftarrow C_0, \ldots, C_n$
    return $C$
```

Fig. 16: The oracles for the games G_1, G_2, G_3, used to prove IND\$-CPLA security of CFB mode

```
adversary $\mathbb{A}_{\mathrm{PRF}}{}^{O}$
    $b' \leftarrow \mathbb{A}^{S(\cdot), \ell[\boldsymbol{\mathcal{E}}]_k(\cdot,\cdot)}$
    return $b'$

simulator $S(M)$
    $M_1, \cdots, M_n \leftarrow M$
    $C_0 \leftarrow\!\$\ \mathsf{T}$
    for $i = 1$ to $n$ do
        $F_i \leftarrow O(C_{i-1})$
        $C_i \leftarrow F_i \star M_i$
    $C \leftarrow C_0, \ldots, C_n$
    return $C$

simulator $\ell[\boldsymbol{\mathcal{E}}]_k(M, l)$
    $M_1, \cdots, M_n \leftarrow M$
    $l_1, \cdots, l_n \leftarrow l$
    $C_0 \leftarrow\!\$\ \mathsf{T}$
    for $i = 1$ to $n$ do
        $F_i, \Lambda_i \leftarrow \ell[\boldsymbol{\mathcal{F}}]_k(C_{i-1}, l_i)$
        $C_i \leftarrow F_i \star M_i$
    $C \leftarrow C_0, \ldots, C_n$
    $\Lambda \leftarrow \Lambda_0, \ldots, \Lambda_n$
    return $(C, \Lambda)$
```

Fig. 17: A PRLF adversary $\mathbb{A}_{\mathrm{PRF}}$ constructed using a distinguisher $\mathbb{A}$ for G_1, G_2, depending if the oracle O is a real or random PRLF call depends which game the adversary $\mathbb{A}$ sees.

Proof (Of Theorem 10). Since CFB mode is a stream cipher, IND–CPLA and IND–aCPLA security coincide if (as in our setting) the leakage sets are identical. Thus we need only prove the IND–CPLA security.

The proof of security follows as a series of game hops between games G_1 and G_3. It can be seen that game G_1 (oracles left of Fig. 16) is exactly the real world of the IND–CPLA game when encryption is instantiated with CFB mode. The game G_3 (oracles right of Fig. 16) is the random world of IND–CPLA. The randomness is generated in blocks for clarity but this does not change the result compared to generating the entire ciphertext at once. Bounding the advantage of an adversary between G_1 and G_3 also bounds the advantage of the adversary winning the IND–CPLA game. To help bound this term we introduce an intermediate game G_2 (oracles middle of Fig. 16) in which the calls to the PRF have been replaced with calls to a truly random function (with memory).

To complete the proof, the advantage of distinguishing G_1 from G_2 and G_2 from G_3 needs to be bounded. An adversary will be able to distinguish G_2 and G_3 only when they can cause a collision in the random function calls, which happens with probability $\frac{\sigma(\sigma-1)^2}{|\mathsf{T}|}$ (by the birthday bound), where σ is the total number of blocks. If an adversary can cause a collision, it means the same value will be input into the random function twice, but since, in game G_3 the random function has no memory, it will return two different values. This will allow the adversary to distinguish between the two games. An adversary who can distinguish between if they are playing game G_1 or game G_2 can be used to distinguish between the real and random world of a PRLF: the reduction is given in Fig. 17. □

To close, we provide justification for the number of generic group elements used, as a function of the number of queries made by the adversary.

Proof (Of Theorems 11 and 6). Apply almost all the previous theorems to oneanother, and collect terms to generate the first bound, given in terms of γ. Thus we are left to bound this quantity, the total number of generic group elements constructed by the adversary. This is just a tedious calculation, and proceeds as follows.

First we note that for each query to the oracle, inputting just arbitrary strings of their own choice they may discover said strings were valid and thus learn three new elements. Thus the g oracle queries made yield up to $3g$ group elements.

Consider now the evaluations of the tagging or PRF function $\mathcal{T}$. Assuming all variables are new, two new elements r must be sampled (2, prior to calling), one hash value (1 operation), 3 partial outputs (3 pairings), 1 actual output (2 operations), and three updated key shares computed (5 operations). Thus in total there are 13 new elements for each call to $\mathcal{T}$.

Suppose the adversary makes q_e encryption queries of total σ_e blocks, and to decryption define q_d, σ_d equivalently. In total across their encryption queries they make q_e PRF queries creating the IVs, σ_e generating the stream, and q_e on tagging queries, along with σ_e new elements from the chaining mode. Thus the total number of elements created during encryption is $13(2q_e + \sigma_e) + \sigma_e$.

On decryption things are slightly more complicated. Consider a PRF call on an input that has already been queried. Such a query repeats the hash call, and the final ciphertext call, thus taking 11 new elements. Verification costs two hash calls more than tagging, but recall invalid queries stop processing at this point. So, invalid queries (which have new content) cost 15 elements. Valid queries do not create so many new elements, but do proceed to complete the whole encryption routine. Thus a valid query requires just 11 new elements per PRF query, but makes as many as the decryption routine. In total then, a valid query of length σ_v creates $11(2 + \sigma_v)$. Thus the number of elements created on a decryption query is maximised by making valid queries, creating a total of $11(2q_d + \sigma_d)$ elements.

Overall, the bound is maximised by making just encryption queries, leading to the claimed bound. □