How to Securely Release Unverified Plaintext in Authenticated Encryption*

Elena Andreeva[1,2], Andrey Bogdanov[3], Atul Luykx[1,2], Bart Mennink[1,2], Nicky Mouha[1,2], and Kan Yasuda[1,4]

[1] Department of Electrical Engineering, ESAT/COSIC, KU Leuven, Belgium.
firstname.lastname@esat.kuleuven.be
[2] iMinds, Belgium.
[3] Department of Mathematics, Technical University of Denmark, Denmark.
anbog@dtu.dk
[4] NTT Secure Platform Laboratories, Japan.
yasuda.kan@lab.ntt.co.jp

Abstract. Scenarios in which authenticated encryption schemes output decrypted plaintext before successful verification raise many security issues. These situations are sometimes unavoidable in practice, such as when devices have insufficient memory to store an entire plaintext, or when a decrypted plaintext needs early processing due to real-time requirements. We introduce the first formalization of the *releasing unverified plaintext* (RUP) setting. To achieve privacy, we propose using *plaintext awareness* (PA) along with IND-CPA. An authenticated encryption scheme is PA if it has a *plaintext extractor*, which tries to fool adversaries by mimicking the decryption oracle, *without* the secret key. Releasing unverified plaintext to the attacker then becomes harmless as it is infeasible to distinguish the decryption oracle from the plaintext extractor. We introduce two notions of plaintext awareness in the symmetric-key setting, PA1 and PA2, and show that they expose a new layer of security between IND-CPA and IND-CCA. To achieve integrity, INT-CTXT in the RUP setting is required, which we refer to as INT-RUP. These new security notions are compared with conventional definitions, and are used to make a classification of symmetric-key schemes in the RUP setting. Furthermore, we re-analyze existing authenticated encryption schemes, and provide solutions to fix insecure schemes.

1 Introduction

The goal of authenticated encryption (AE) is to simultaneously provide data privacy and integrity. AE decryption conventionally consists of two phases: plaintext computation and verification. As reflected in classical security models, plaintext coming from decryption is output only upon successful verification.

* This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and OT/13/071. Elena Andreeva, Bart Mennink, and Nicky Mouha are Postdoctoral Fellows of the Research Foundation – Flanders (FWO). Atul Luykx is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

Nevertheless, there are settings where releasing plaintext before verification is desirable. For example, it is necessary if there is not enough memory to store the entire plaintext [21] or because real-time requirements would otherwise not be met [15, 38]. Even beyond these settings, using dedicated schemes secure against the release of unverified plaintext can increase efficiency. For instance, to avoid releasing unverified plaintext into a device with insecure memory [37], the two-pass Encrypt-then-MAC composition can be used: a first pass to verify the MAC, and a second to decrypt the ciphertext. However, a single pass AE scheme suffices if it is secure against the release of unverified plaintext.

If the attacker cannot observe the unverified plaintext directly, it may be possible to determine properties of the plaintext through a side channel. This occurs, for example, in the padding oracle attacks introduced by Vaudenay [39], where an error message or the lack of an acknowledgment indicates whether the unverified plaintext was correctly padded. Canvel et al. [18] showed how to mount a padding oracle attack on the then-current version of OpenSSL by exploiting timing differences in the decryption processing of TLS. As shown by Paterson and AlFardan [1, 30] for TLS and DTLS, it is very difficult to prevent an attacker from learning the cause of decryption failures.

The issue of releasing unverified plaintext has also been acknowledged and explicitly discussed in the upcoming CAESAR competition:[5] *"Beware that security questions are raised by any authenticated cipher that handles a long ciphertext in one pass without using a large buffer: releasing unverified plaintext to applications often means releasing it to attackers and also requires an analysis of how the applications will react."*

For several AE schemes, including OCB [27], AEGIS [41], ALE [15], and FIDES [15], the designers explicitly stress that unverified plaintext cannot be released. Although the issue of *releasing unverified plaintext* (RUP) in AE is frequently discussed in the literature, it has largely remained unaddressed even in recent AE proposals, likely due to a lack of comprehensive study.

We mention explicitly that we do *not* recommend omitting verification, which remains essential to preventing incorrect plaintexts from being accepted. To ensure maximal security, unverified plaintext must be kept hidden from adversaries. However, our scenario assumes that the attacker can see the unverified plaintext, or any information relating to it, before verification is complete. Furthermore, issues related to the behavior of applications which process unverified plaintext are beyond the scope of this paper; careful analysis is necessary in such situations.

1.1 Security Under Release of Unverified Plaintext

AE security is typically examined using indistinguishability under chosen plaintext attack (IND-CPA) for privacy and integrity of ciphertexts (INT-CTXT) for integrity, and a scheme which achieves both is indistinguishable under chosen ciphertext attack (IND-CCA), as shown by Bellare and Namprempre [9] and Katz and Yung [26]. However, in the RUP situation adversaries can also observe

[5] `http://competitions.cr.yp.to/features.html`

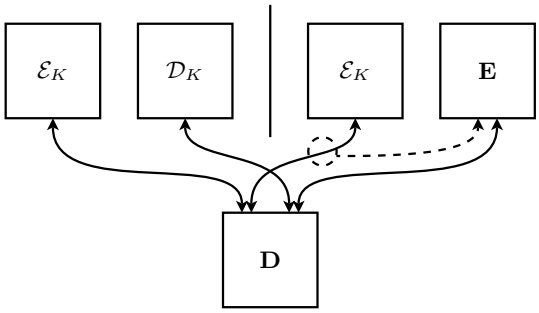

Fig. 1. The two plaintext aware settings (PA1 and PA2) used in the paper, where **D** is an adversary. Not shown in the figure is the type of IV used by the $\mathcal{E}_K$ oracle (cf. Sect. 3.2). **Left:** Real world, with encryption oracle $\mathcal{E}_K$ and decryption oracle $\mathcal{D}_K$. **Right:** Simulated world, with encryption oracle $\mathcal{E}_K$ and plaintext extractor **E**. The plaintext extractor **E** is a stateful algorithm without knowledge of the secret key K, nor access to the encryption oracle $\mathcal{E}_K$. The dotted line indicates that **E** has access to the encryption queries made by adversary **D**, which only holds in the PA1 setting.

unverified plaintext, which the conventional definitions do not take into account. To address this gap we introduce two definitions: integrity under releasing unverified plaintext (INT-RUP) and *plaintext awareness* (PA). For integrity we propose using INT-RUP and for privacy both IND-CPA and PA. In the full version of this paper [3], we discuss how the combination of INT-RUP, IND-CPA, and PA measures the impact of releasing unverified plaintext on security.

INT-RUP. The goal of an adversary under INT-CTXT is to produce new ciphertexts which pass verification, with only access to the encryption oracle. We translate INT-CTXT into the RUP setting, called INT-RUP, by allowing the adversary to observe unverified plaintexts. We formalize this by separating plaintext computation from verification, and giving the adversary access to a plaintext-computing oracle.

Plaintext Awareness (PA). We introduce PA as a new symmetric-key notion to achieve security in the RUP setting. Informally, we define a scheme to be PA if the adversary cannot gain any additional knowledge about the plaintext from decryption queries besides what it can derive from encryption queries.

Our PA notion only involves encryption and decryption, and can thus be defined both for encryption schemes as well as for AE schemes that release unverified plaintext.

At the heart of our new PA notion is the *plaintext extractor*, shown in Fig. 1. We say that an encryption scheme is PA if it has an efficient plaintext extractor, which is a stateful algorithm that mimicks the decryption oracle in order to fool the adversary. It cannot make encryption nor decryption queries, and does not know the secret key. We define two notions of plaintext awareness: PA1 and PA2. The extractor is given access to the history of queries made to the encryption oracle in PA1, but not in PA2. Hence PA1 is used to model RUP scenarios in which the goal of the adversary is to gain knowledge beyond what it knows from

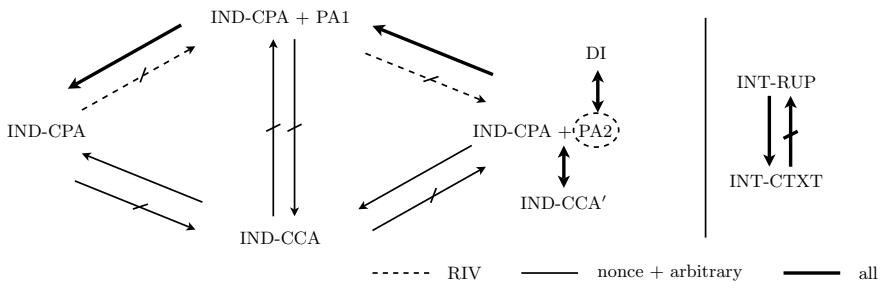

Fig. 2. Implications and separations between the IND-CPA, IND-CPA+PA1, IND-CPA+PA2, IND-CCA, IND-CCA′, PA2, and DI security notions (left) and INT-CTXT and INT-RUP (right). Dashed lines refer to relations that hold if the IV is random and thin solid lines in case of nonce or arbitrary IV. We use a thick solid line if the relation holds under all IV cases.

the query history. For situations in which the goal of the adversary is to decrypt one of the ciphertexts in the query history, we require PA2.

Relations Among Notions. PA for public-key encryption was introduced by Bellare and Rogaway [11], and later defined without random oracles by Bellare and Palacio [10]. In the symmetric-key setting, our definition of PA is somewhat similar, however there are important technical differences which make the public-key results inapplicable to the symmetric-key setting.

Relations among the PA and conventional security definitions for encryption (see Sect. 3.3) are summarized in Fig. 2. We consider three IV assumptions: random IV, nonce IV (non-repeating value), and arbitrary IV (value that can be reused), as explained in Sect. 3.2. The statements of the theorems and proofs can be found in the full version of this paper [3].

The motivation for having two separate notions, PA1 and PA2, is as follows. As we prove in this work, if the plaintext extractor has access to the query history (PA1), then there are no implications between IND-CPA+PA1 and IND-CCA. However, if we modify plaintext awareness so that the plaintext extractor no longer has access to the query history (PA2), then we can prove that IND-CPA+PA2 implies IND-CCA′. IND-CCA′ is a strengthened version of IND-CCA, where we allow the adversary to re-encrypt the outputs of the decryption oracle. Note that such a re-encryption is always allowed in the public-key setting, but not in the symmetric-key setting where the key required for encryption is secret. Furthermore, we also prove that PA2 is equivalent to the notion of *decryption independence* (DI). DI captures the fact that encryption and decryption under the same key are only related to each other as much as encryption and decryption under different keys.

Finally, although INT-RUP clearly implies INT-CTXT, the opposite is not necessarily true.

Motivating Examples. To get an intuition for PA1 (shown in Fig. 1) and how it relates to the RUP setting, we provide two motivating examples with CTR mode. For simplicity, we define the encryption function of CTR mode as

$\mathcal{E}_K(\mathrm{IV}, M) = E_K(\mathrm{IV}) \oplus M$, where the message M and the initialization value IV consist of one block each, and E_K is a block cipher with a secret key K. The corresponding decryption function is $\mathcal{D}_K(\mathrm{IV}, C) = E_K(\mathrm{IV}) \oplus C$. As shown in [13], CTR mode is IND-CPA but not IND-CCA, a result that holds for nonce IVs (unique non-repeating values) as well as for random IVs.

1. *Nonce IV CTR mode is not PA1.* Following Rogaway [31], we assume that an adversary is free to specify the IV for encryption and decryption queries, as long as it does not make two encryption queries with the same nonce IV, N. In the attack, an adversary first makes a decryption query (N, C) with nonce N and one-block ciphertext C to obtain a message M. The correct decryption of M is $E_K(N) \oplus C$ as output by the decryption oracle. The adversary then computes the keystream $\kappa := M \oplus C$. Now in a second query (N, M'), this time to the encryption oracle, the adversary obtains C' where $C' = M' \oplus \kappa$.
   The scheme fails to be plaintext aware as it is infeasible for any plaintext extractor to be consistent with subsequent encryption queries. Specifically, the plaintext extractor cannot compute κ at the time of the first decryption query for the following reasons: it does not know the secret key K, it is not allowed to do encryption queries, and an encryption query with N has not yet been recorded in the query history.
2. *Random IV CTR mode is PA1.* In this setting, the IV used in encryption is chosen randomly by the environment, and therefore out of the attacker's control. However, the adversary can still freely choose the IV for its decryption queries. In this random IV setting, the attack in the nonce IV example does not apply. To see this, consider an adversary which queries the decryption of (IV_1, C) with a one-block ciphertext C. It can compute the keystream associated to IV_1, but does not control when IV_1 is used in encryption. Thus, a plaintext extractor can be defined as outputting a random plaintext M in response to the (IV_1, C) query.
   But what if an adversary makes additional decryption queries with the same IV? Suppose the adversary makes decryption query $(\mathrm{IV}_1, C \oplus \Delta)$. Since the plaintext extractor is a stateful algorithm, it can simply output $M \oplus \Delta$ to provide consistency. Furthermore, if an adversary makes encryption queries, these will be seen by the PA1 plaintext extractor. Therefore, the plaintext extractor can calculate the keystream from these queries, and respond to any decryption queries in a consistent way. A proof that random IV CTR mode is PA1 is provided in Prop. 2.

AE schemes such as GCM [28] and CCM [40] reduce to CTR mode in the RUP setting. This is because the adversary does not need to forge a ciphertext in order to obtain information about the corresponding (unverified) plaintext. By requiring that the underlying encryption scheme of an AE scheme is PA1, we ensure that the adversary does not gain any information from decryption queries, meaning no decryption query can be used to find an inconsistency with any past or future queries to the encryption or decryption oracles.

1.2 Analysis of Authenticated Encryption Schemes

Given the formalization of AE in the RUP setting, we categorize existing AE schemes based on the type of IV used by the encryption function: random IV, nonce IV, and arbitrary IV. Then, we re-analyze the security of several recently proposed AE schemes as well as more established AE schemes. In order to do so, we split the decryption algorithms into two parts, plaintext computation and verification, as described in Sect. 3.1.

For integrity, we show that OCB [33] and COPA [4] succumb to attacks by using unverified plaintext to construct forgeries. For privacy an overview of our results can be seen in Table 1, where we also include the encryption-only modes CTR and CBC as random IV examples. We draw a distinction between the schemes that are *online* and the schemes that are not, where an online scheme is one that is able to produce ciphertext blocks as it receives plaintext blocks.

Most of the schemes in Table 1 fail to achieve PA1. As a result, we demonstrate techniques to restore PA1 for nonce IV and arbitrary IV schemes. For the former, we introduce the *nonce decoy* technique, and for the latter the PRF-to-IV method, which converts a random IV PA1 scheme into an arbitrary IV PA1 scheme. For online arbitrary IV schemes, we demonstrate that PA1 security can be achieved only if the ciphertext is substantially longer than the plaintext, or the *decryption* is offline. We show that McOE-G [20] achieves PA1 if the plaintext is padded so that the ciphertext becomes twice as long. We also prove that APE [2], an online deterministic AE scheme with offline decryption, achieves PA1.

Finally we show that the nonce decoy preserves INT-RUP, and the PRF-to-IV method turns any random IV scheme into an INT-RUP arbitrary IV scheme.

1.3 Background and Related Work

The definition of encryption and AE has been extended and generalized in different ways. In 2004, Rogaway [32] introduced nonce IV encryption schemes, in contrast with prior encryption modes that used a random IV, as in the CBC mode standardized by NIST in 1980 [29].

Rogaway and Shrimpton [34] formalized deterministic AE (DAE), where an IV input is optional and can therefore take arbitrary values. Secure DAE differs from secure nonce IV AE schemes in that DAE privacy is possible only up to message repetition, namely an adversary can detect repeated encryptions. Unfortunately, DAE schemes cannot be online. To resolve this issue, Fleischmann et al. [20] explored online DAE schemes, where privacy holds only up to repetitions of messages with identical prefixes or up to the longest common prefix.

Tsang et al. [38] gave syntax and security definitions of AE for streaming data. Bellare and Keelveedhi [6] considered a stronger security model where data may be key-dependent. Boldyreva et al. reformulated AE requirements and properties to handle ciphertext fragmentation in [16], and enhanced the syntax and security definitions so that the verification oracle is allowed to handle multiple failure events in [17]. Our formalization can be interpreted as a special case of the work in [17], yet the emphasis and results differ.

Table 1. PA1 and PA2 security of deterministic and non-deterministic schemes, separated as described in Sect. 3.1. In the columns for PA1 and PA2, ✓ means secure (there exists an extractor), and ✗ means insecure (there exists an attack). Proofs for the security results in this table can be found in Sect. 5.

IV type	Online	Scheme	PA1	PA2	Remark
random	✓	CTR, CBC [29]	✓	✗	
nonce	✓	OCB [33]	✗	✗	
	✓	GCM [28], SpongeWrap [14]	✗	✗	
	✗	CCM [40]	✗	✗	not online [35]
arbitrary	✓	COPA [4]	✗	✗	privacy up to prefix
	✓	McOE-G [20]	✗	✗	〃
	✓	APE [2]	✓	✗	〃, backwards decryption
	✗	SIV [34], BTM [23], HBS [24]	✓	✗	privacy up to repetition
	✗	Encode-then-Encipher [12]	✓	✓	〃, VIL SPRP, padding

2 Preliminaries

Symbols. Given two strings A and B in $\{0,1\}^*$, we use $A\|B$ and AB interchangeably to denote the concatenation of A and B. The symbol $\oplus$ denotes the bitwise XOR operation of two strings. Addition modulo 2^n is denoted by $+$, where n usually is the bit length of a block. For example, in the CTR mode of operation of a block cipher, we increment the IV value by addition $IV+i \pmod{2^n}$, where n is the block size, the n-bit string $IV = IV_{n-1}\cdots IV_1 IV_0 \in \{0,1\}^n$ is converted to an integer $2^{n-1}IV_{n-1}+\cdots+2IV_1+IV_0 \in \{0,1,\ldots,2^n-1\}$, and the result of addition is converted to an n-bit string in the reverse way. By $K \xleftarrow{R} \mathsf{K}$ we mean that K is chosen uniformly at random from the set K. All algorithms and adversaries are considered to be "efficient".

Adversaries and Advantages. An adversary is an oracle Turing machine. Let $\mathbb{D}$ be some class of computationally bounded adversaries; a class $\mathbb{D}$ can consist of a single adversary $\mathbf{D}$, i.e. $\mathbb{D} = \{\mathbf{D}\}$, in which case we simply write $\mathbf{D}$ instead of $\mathbb{D}$. For convenience, we use the notation

$$\underset{\mathbb{D}}{\Delta}(f\,;\,g) := \sup_{\mathbf{D}\in\mathbb{D}} \left|\Pr[\mathbf{D}^f = 1] - \Pr[\mathbf{D}^g = 1]\right|$$

to denote the supremum of the distinguishing advantages over all adversaries distinguishing oracles f and g, where the notation $\mathbf{D}^{\mathcal{O}}$ indicates the value output by $\mathbf{D}$ after interacting with oracle $\mathcal{O}$. The probabilities are defined over the random coins used in the oracles and the random coins of the adversary, if any. Multiple oracles are separated by a comma, e.g. $\Delta(f_1, f_2\,;\,g_1, g_2)$ denotes distinguishing the combination of f_1 and f_2 from the combination of g_1 and g_2.

If $\mathbf{D}$ is distinguishing $(f_1, f_2, \ldots, f_k)$ from $(g_1, g_2, \ldots, g_k)$, then by $\mathcal{O}_i$ we mean the ith oracle that $\mathbf{D}$ has access to, i.e. either f_i or g_i depending upon which oracles it is interacting with. By $\mathcal{O}_i \hookrightarrow \mathcal{O}_j$ we describe a set of actions that $\mathbf{D}$ can perform: first $\mathbf{D}$ queries $\mathcal{O}_i$, and then at some point in the future $\mathbf{D}$

queries $\mathcal{O}_j$ with the output of $\mathcal{O}_i$, assuming the output of $\mathcal{O}_i$ can be used directly as the input for $\mathcal{O}_j$. If the oracles $\mathcal{O}_i$ and $\mathcal{O}_j$ represent a family of algorithms indexed by inputs, then the indices must match. For example, say that $\mathcal{E}_K^{N,A}$ and $\mathcal{D}_K^{N,A}$ are families indexed by (N, A). Then $\mathcal{E}_K \hookrightarrow \mathcal{D}_K$ describes a set of actions, which includes querying $\mathcal{E}_K^{N,A}(M)$ to receive C, and then at some point in the future querying $\mathcal{D}_K^{N,A}(C)$, where K, N, A, and C are reused.

Our security definitions follow [9] and are given in terms of adversary advantages. A scheme is said to be secure with respect to some definition if it is negligible with respect to all adversaries with time complexity polynomial in the security parameter. As in [9], positive results are given as explicit bounds, whereas negative results, i.e. separations, are given in asymptotic terms, which can easily be converted into concrete bounds.

Online Functions. A function $f : \mathsf{M} \to \mathsf{C}$ is said to be *n-online* if there exist functions $f_i : \{0,1\}^i \to \{0,1\}^{c_i}$ and $f'_i : \{0,1\}^i \to \{0,1\}^{c'_i}$ such that $c_i > 0$, and for all $M \in \mathsf{M}$ we have

$$f(M) = f_n(M_1)\, f_{2n}(M_1 M_2) \cdots f_{jn}(M_1 M_2 \cdots M_j)\, f'_{|M|}(M) \ ,$$

where $j = \lfloor (|M| - 1)/n \rfloor$ and M_i is the ith n-bit block of M. Often we just say f is *online* if the value n is clear from context.

3 AE Schemes: Syntax, Types, and Security

3.1 New AE Syntax

A conventional AE scheme $\Pi = (\mathcal{E}, \mathcal{D})$ consists of an encryption algorithm $\mathcal{E}$ and a decryption algorithm $\mathcal{D}$:

$$(C, T) \leftarrow \mathcal{E}_K^{IV,A}(M) \ ,$$
$$M/\perp \leftarrow \mathcal{D}_K^{IV,A}(C, T) \ ,$$

where $K \in \mathsf{K}$ is a key, $IV \in \mathsf{IV}$ an initialization value, $A \in \mathsf{A}$ associated data, $M \in \mathsf{M}$ a message, $C \in \mathsf{C}$ the ciphertext, $T \in \mathsf{T}$ the tag, and each of these sets is a subset of $\{0,1\}^*$. The *correctness condition* states that for all K, IV, A, and M, $\mathcal{D}_K^{IV,A}(\mathcal{E}_K^{IV,A}(M)) = M$. A secure AE scheme should return $\perp$ when it does not receive a valid (C, T) tuple.

In order to consider what happens when unverified plaintext is released, we disconnect the decryption algorithm from the verification algorithm so that the decryption algorithm always releases plaintext. A separated AE scheme is a triplet $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{V})$ of keyed algorithms — encryption $\mathcal{E}$, decryption $\mathcal{D}$, and verification $\mathcal{V}$ — such that

$$(C, T) \leftarrow \mathcal{E}_K^{IV,A}(M) \ ,$$
$$M \leftarrow \mathcal{D}_K^{IV,A}(C, T) \ ,$$
$$\top/\perp \leftarrow \mathcal{V}_K^{IV,A}(C, T) \ ,$$

where K, IV, A, M, C, and T are defined as above. Note that in some deterministic schemes IV may be absent, in which case we can expand the interface of such schemes to receive IV input with which it does nothing. Furthermore, for simplicity we might omit A if there is no associated data. The special symbols $\top$ and $\bot$ indicate the success and failure of the verification process, respectively.

As in the conventional setting we impose a *correctness condition*: for all K, IV, A, and M such that $\mathcal{E}_K^{IV,A}(M) = (C, T)$, we require $\mathcal{D}_K^{IV,A}(C, T) = M$ and $\mathcal{V}_K^{IV,A}(C, T) = \top$.

Relation to Conventional Syntax. Given a separated AE scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{V})$, we can easily convert it into a conventional AE scheme $\overline{\Pi} = (\mathcal{E}, \overline{\mathcal{D}})$. Remember that the conventional decryption oracle $\overline{\mathcal{D}}_K^{IV,A}(C, T)$ outputs M where $M = \mathcal{D}_K^{IV,A}(C, T)$ if $\mathcal{V}_K^{IV,A}(C, T) = \top$, and $\bot$ otherwise.

The conversion in the other direction is not immediate. While the verification algorithm $\mathcal{V}$ can be easily "extracted" from $\overline{\mathcal{D}}$ (i.e., one can easily construct $\mathcal{V}$ using $\overline{\mathcal{D}}$ — just replace M with $\top$), it is not clear if one can always "naturally" extract the decryption algorithm $\mathcal{D}$ from $\overline{\mathcal{D}}$. However, all practical AE schemes that we are aware of can be constructed from a triplet $(\mathcal{E}, \mathcal{D}, \mathcal{V})$ as above, and hence their decryption algorithms $\overline{\mathcal{D}}$ are all separable into $\mathcal{D}$ and $\mathcal{V}$.

3.2 Types of AE Schemes

Classification Based on IVs. In order to achieve semantic security [22], AE schemes must be probabilistic or stateful [5]. Usually the randomness or state is focused into an IV [32]. How the IV is used restricts the scheme's syntax and the types of adversaries considered in the security notions:

1. **Random IV.** The environment chooses a random IV for each encryption, thus an adversary has no control over the choice of IV for each encryption. The generated IV must be sent along with the ciphertext so that the receiver can decrypt.
2. **Nonce IV.** A distinct IV must be used for each encryption, thus an adversary can choose but does not repeat nonce IV values in its encryption queries. How the parties synchronize the nonce is typically left implicit.
3. **Arbitrary IV.** No restrictions on the IV are imposed, thus an adversary may choose any IV for encryption. Often a deterministic AE scheme does not even have an IV input, in which case an IV can be embedded into the associated data A, which gets authenticated along with the plaintext M but does not get encrypted; A is sent in the clear.

In all IV cases the adversary can arbitrarily choose the IV input values to the decryption oracle. In some real-world protocols the decryption algorithm can be stateful [7], but such schemes are out of the scope of this paper, and schemes designed to be secure with deterministic decryption algorithms will be secure in those settings as well.

While random and nonce IV schemes can achieve semantic security, arbitrary IV schemes cannot, and therefore reduce to deterministic security. In the

Table 2. The type of random oracle needed depending upon the class of AE scheme considered.

IV type	type of encryption	
	online	offline
random	random oracle	random oracle
nonce	random oracle	random oracle
arbitrary	random-up-to-prefix oracle	random-up-to-repetition oracle

latter case, the most common notions are "*privacy up to repetition*" which is used for DAE [34] and "*privacy up to prefix*" which is used for authenticated online encryption [20]. In any case, we write \$ to indicate the ideal oracle from which an adversary tries to distinguish the real encryption oracle $\mathcal{E}_K$, regardless of the IV type. This means that the ideal \$ oracle should be either the random oracle, random-up-to-repetition oracle, or random-up-to-prefix oracle, depending upon the IV. Each of the cases with their respective random oracles are listed in Table 2. In order to avoid redundancy in the wording of the definitions, whenever we write $\Delta(\mathcal{E}_K, \ldots ; \$, \ldots)$, it is understood that the \$ oracle is the one appropriate for the AE scheme consisting of $\mathcal{E}$.

Online Encryption/Decryption Algorithms. A further distinction is made between online schemes and the others. An AE scheme with online encryption is one in which the ciphertext can be output as the plaintext is received, namely we require that for each (K, IV, A) the resulting encryption function is online as a function of the plaintext M.

Although decryption in AE schemes can never be online due to the fact that the message needs to be verified before it is output, we still consider schemes which can compute the plaintext as the ciphertext is received. In particular, a scheme with online decryption is one in which this plaintext-computing algorithm, viewed as a function of the ciphertext and tag input, is online. Note that in some schemes the tag could be received before the ciphertext, in which case we still consider $\mathcal{D}$ to be online (even though our new syntax implies that the tag is always received after the ciphertext).

3.3 Conventional Security Definitions under the New Syntax

Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{V})$ denote an AE scheme as a family of algorithms indexed by the key, IV, and associated data. With the new separated syntax we reformulate the conventional security definitions, IND-CPA, IND-CCA, and INT-CTXT. As mentioned above, the security notions are defined in terms of an unspecified \$, where the exact nature of \$ depends on the type of IV allowed (cf. Table 2). In the definitions the only fixed input to the algorithms is the key, indicated by writing $\mathcal{E}_K$ and $\mathcal{D}_K$; all other inputs, such as the IV and associated data, can be entered by the adversary.

Definition 1 (IND-CPA Advantage). *Let* $\mathbf{D}$ *be a computationally bounded adversary with access to one oracle* $\mathcal{O}$*, and let* $K \xleftarrow{R} \mathsf{K}$*. Then the IND-CPA*

advantage of $\mathbf{D}$ *relative to* Π *is given by*

$$\mathsf{CPA}_{\Pi}(\mathbf{D}) := \underset{\mathbf{D}}{\Delta}(\mathcal{E}_K\,;\,\$)\ .$$

Definition 2 (IND-CCA Advantage). *Let* $\mathbf{D}$ *be a computationally bounded adversary with access to two oracles* $\mathcal{O}_1$ *and* $\mathcal{O}_2$*, such that* $\mathbf{D}$ *never queries* $\mathcal{O}_1 \hookrightarrow \mathcal{O}_2$ *nor* $\mathcal{O}_2 \hookrightarrow \mathcal{O}_1$*, and let* $K \xleftarrow{R} \mathsf{K}$*. Then the IND-CCA advantage of* $\mathbf{D}$ *relative to* Π *is given by*

$$\mathsf{CCA}_{\Pi}(\mathbf{D}) := \underset{\mathbf{D}}{\Delta}(\mathcal{E}_K, \mathcal{D}_K\,;\,\$, \mathcal{D}_K)\ .$$

Note that IND-CCA as defined above does not apply to the random IV setting. When a random IV is used, the adversary is not prohibited from querying $\mathcal{O}_2 \hookrightarrow \mathcal{O}_1$. We introduce a version of IND-CCA below, which can be applied to all IV settings.

Definition 3 (IND-CCA′ Advantage). *Let* $\mathbf{D}$ *be an adversary as in Def. 2, except* $\mathbf{D}$ *may now query* $\mathcal{O}_2 \hookrightarrow \mathcal{O}_1$*, and let* $K \xleftarrow{R} \mathsf{K}$*. Then the IND-CCA′ advantage of* $\mathbf{D}$ *relative to* Π *is given by*

$$\mathsf{CCA}'_{\Pi}(\mathbf{D}) := \underset{\mathbf{D}}{\Delta}(\mathcal{E}_K, \mathcal{D}_K\,;\,\$, \mathcal{D}_K)\ .$$

Definition 4 (INT-CTXT Advantage). *Let* $\mathbf{F}$ *be a computationally bounded adversary with access to two oracles* $\mathcal{E}_K$ *and* $\mathcal{V}_K$*, such that* $\mathbf{F}$ *never queries* $\mathcal{E}_K \hookrightarrow \mathcal{V}_K$*. Then the INT-CTXT advantage of* $\mathbf{F}$ *relative to* Π *is given by*

$$\mathsf{CTXT}_{\Pi}(\mathbf{F}) := \Pr\left[\mathbf{F}^{\mathcal{E}_K, \mathcal{V}_K} \text{ forges}\right]\ ,$$

where the probability is defined over the random key K *and random coins of* $\mathbf{F}$*. Here,* "forges" *means that* $\mathcal{V}_K$ *returns* $\top$ *to the adversary.*

4 Security Under Release of Unverified Plaintext

4.1 Security of Encryption

We introduce the notion of plaintext-aware encryption of symmetric-key encryption schemes. An analysis of existing plaintext-aware schemes can be found in Sect. 5. The formalization is similar to the one in the public-key setting [10]. Let $\Pi = (\mathcal{E}, \mathcal{D})$ denote an encryption scheme.

Definition 5 (PA1 Advantage). *Let* $\mathbf{D}$ *be an adversary with access to two oracles* $\mathcal{O}_1$ *and* $\mathcal{O}_2$*. Let* $\mathbf{E}$ *be an algorithm with access to the history of queries made to* $\mathcal{O}_1$ *by* $\mathbf{D}$*, called a* PA1-extractor*. We allow* $\mathbf{E}$ *to maintain state across invocations. The PA1 advantage of* $\mathbf{D}$ *relative to* $\mathbf{E}$ *and* Π *is*

$$\mathsf{PA1}^{\mathbf{E}}_{\Pi}(\mathbf{D}) := \underset{\mathbf{D}}{\Delta}(\mathcal{E}_K, \mathcal{D}_K\,;\,\mathcal{E}_K, \mathbf{E})\ ,$$

where $K \xleftarrow{R} \mathsf{K}$*, and the probability is defined over the key* K*, the random coins of* $\mathbf{D}$*, and the random coins of* $\mathbf{E}$*.*

The adversary $\mathbf{D}$ tries to distinguish the case in which its second oracle $\mathcal{O}_2$ is given by $\mathcal{D}_K$ versus the case in which $\mathcal{O}_2$ is given by $\mathbf{E}$. The task of $\mathbf{E}$ is to mimic the outputs of $\mathcal{D}_K$ given only the history of queries made to $\mathcal{E}_K$ by $\mathbf{D}$ (the key is not given to $\mathbf{E}$). Note that $\mathbf{D}$ is allowed to make queries of the form $\mathcal{E}_K \hookrightarrow \mathbf{E}$; these can easily be answered by $\mathbf{E}$ via the query history.

PA2 is a strengthening of PA1 where the extractor no longer has access to the query history of $\mathcal{E}_K$; the extractor becomes a simulator for the decryption algorithm. Note that in order for this to work, we cannot allow the adversaries to make queries of the form $\mathcal{E}_K \hookrightarrow \mathbf{E}$.

Definition 6 (PA2 Advantage). *Let* $\mathbf{D}$ *be an adversary as in Def. 5, with the added restriction that it may not ask queries of the form* $\mathcal{O}_1 \hookrightarrow \mathcal{O}_2$*. Let* $\mathbf{E}$ *be an algorithm, called a* PA2-extractor*. We allow* $\mathbf{E}$ *to maintain state across invocations. The PA2 advantage of* $\mathbf{D}$ *relative to* $\mathbf{E}$ *and* Π *is*

$$\mathsf{PA2}^{\mathbf{E}}_{\Pi}(\mathbf{D}) := \underset{\mathbf{D}}{\Delta}(\mathcal{E}_K, \mathcal{D}_K \,;\, \mathcal{E}_K, \mathbf{E}) \ ,$$

where $K \xleftarrow{R} \mathsf{K}$*, and the probability is defined over the key* K*, the random coins of* $\mathbf{D}$*, and the random coins of* $\mathbf{E}$*.*

An equivalent way of describing PA2 is via *decryption independence* (DI), which means that the adversary cannot distinguish between encryption and decryption under the same key and under different keys. The equivalence between PA2 and DI is proven in [3].

Definition 7 (Decryption Independence). *Let* $\mathbf{D}$ *be a distinguisher accepting two oracles not making queries of the form* $\mathcal{O}_1 \hookrightarrow \mathcal{O}_2$*, then the DI advantage of* $\mathbf{D}$ *relative to* Π *is*

$$\mathsf{DI}_{\Pi}(\mathbf{D}) := \underset{\mathbf{D}}{\Delta}(\mathcal{E}_K, \mathcal{D}_K \,;\, \mathcal{E}_K, \mathcal{D}_L) \ ,$$

where $K, L \xleftarrow{R} \mathsf{K}$ *are independent.*

4.2 Security of Verification

Integrity when releasing unverified plaintext is a modification of INT-CTXT (Def. 4) to include the decryption oracle as a means to obtain unverified plaintext. Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{V})$ be an AE scheme with separate decryption and verification.

Definition 8 (INT-RUP Advantage). *Let* $\mathbf{F}$ *be a computationally bounded adversary with access to three oracles* $\mathcal{E}_K$*,* $\mathcal{D}_K$*, and* $\mathcal{V}_K$*, such that* $\mathbf{F}$ *never queries* $\mathcal{E}_K \hookrightarrow \mathcal{V}_K$*. Then the INT-RUP advantage of* $\mathbf{F}$ *relative to* Π *is given by*

$$\mathsf{INT\text{-}RUP}_{\Pi}(\mathbf{F}) := \Pr\left[\mathbf{F}^{\mathcal{E}_K, \mathcal{D}_K, \mathcal{V}_K} \text{ forges}\right] \ ,$$

where the probability is defined over the key K *and random coins of* $\mathbf{F}$*. Here,* "forges" *means the event of the oracle* $\mathcal{V}_K$ *returning* $\top$ *to the adversary.*

5 Achieving Plaintext Awareness

5.1 Why Existing Schemes Do Not Achieve PA1

In conventional AE schemes such as OCB, GCM, SpongeWrap, CCM, COPA, and McOE-G, a ciphertext is computed using some bijective function, and then a tag is appended to the ciphertext. The schemes achieve AE because the tag prevents all ciphertexts from being valid. But if the tag is no longer checked, then we cannot achieve PA1, as explained below.

Let $\Pi = (\mathcal{E}_K, \mathcal{D}_K)$ be a nonce or arbitrary IV encryption scheme, then we can describe Π as follows,

$$\mathcal{E}_K^{IV,A}(M) = E_K^{IV,A}(M) \parallel F_K^{IV,A}(M) \ ,$$

where E_K is length-preserving, i.e. $|E_K^{IV,A}(M)| = |M|$. One can view $F_K^{IV,A}(M)$ as the tag-producing function from a scheme such as GCM. In the following proposition we prove that if Π is IND-CPA and PA1, then E_K cannot be bijective for each (IV, A), assuming either a nonce or arbitrary IV. Note that the proposition only holds if Π is a nonce or arbitrary IV scheme.

Proposition 1. *Say that E_K is bijective for all (IV, A), then there exists an adversary $\mathbf{D}$ such that for all extractors $\mathbf{E}$, there exists an adversary $\mathbf{D}_1$ such that*

$$1 - \mathsf{CPA}_{\Pi}(\mathbf{D}_1) \leq \mathsf{PA1}_{\Pi}^{\mathbf{E}}(\mathbf{D}) \ ,$$

where $\mathbf{D}$ makes one $\mathcal{O}_1$ query, one $\mathcal{O}_2$ query, and $\mathbf{D}_1$ is as efficient as $\mathbf{D}$ plus one query to $\mathbf{E}$.

Proof. See [3]. □

We conclude that in order for a nonce or arbitrary IV scheme to be PA1 and IND-CPA, E_K must either not be bijective, or not be length-preserving.

5.2 PA1 Random IV Schemes

We illustrate Def. 5 and the idea of an extractor by considering the CTR mode with a random IV.

Example 1 (RIV-CTR Extractor). Let $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a PRF. For $M_i \in \{0,1\}^n$, $1 \leq i \leq \ell$, define RIV-CTR encryption as

$$\mathcal{E}_K^{C_0}(M_1 \cdots M_\ell) = F_K(C_0 + 1) \oplus M_1 \parallel \cdots \parallel F_K(C_0 + \ell) \oplus M_\ell \ ,$$

where C_0 is selected uniformly at random from $\{0,1\}^n$ for each encryption, and decryption as

$$\mathcal{D}_K^{C_0}(C_1 \cdots C_\ell) = F_K(C_0 + 1) \oplus C_1 \parallel \cdots \parallel F_K(C_0 + \ell) \oplus C_\ell \ .$$

We can define an extractor $\mathbf{E}$ for RIV-CTR as follows. Initially, $\mathbf{E}$ generates a random key K' which it will use via $F_{K'}$. Let $(C_0, C_1 \cdots C_\ell)$ denote an input to $\mathbf{E}$. Using C_0, the extractor searches its history for a ciphertext with C_0 as IV.

1. If such a ciphertext exists, we let $(C'_1 \cdots C'_m, M'_1 \cdots M'_m)$ denote the longest corresponding $\mathcal{E}_K$ query-response pair. Define $\kappa_i := C'_i \oplus M'_i$ for $1 \leq i \leq \min\{\ell, m\}$. Notice that κ_i corresponds to the keystream generated by F_K for $1 \leq i \leq \ell$. For $m < i \leq \ell$ we generate κ_i by $F_{K'}(C_0 + i)$.
2. If there is no such ciphertext, then we generate κ_i as $F_{K'}(C_0+i)$ for $1 \leq i \leq \ell$.

Then we set $\mathbf{E}^{C_0}(C_1 \cdots C_\ell) = (C_1 \oplus \kappa_1, C_2 \oplus \kappa_2 \parallel \cdots \parallel C_\ell \oplus \kappa_\ell)$.

Proposition 2. *Let* $\mathbf{D}$ *be a PA1 adversary for RIV-CTR making queries whose lengths in number of blocks sum up to* σ*, then*

$$\mathsf{PA1}^{\mathbf{E}}_{RIV\text{-}CTR}(\mathbf{D}) \leq \underset{\mathbf{D}_1}{\Delta}(F_K, F_K\,;\, F_K, F_{K'}) + \frac{\sigma^2}{2^n} \ ,$$

where $\mathbf{D}_1$ *is an adversary which may not make the same query to both of its oracles, and makes a total of* σ *queries with the same running time as* $\mathbf{D}$*.*

We refer to a proof of this proposition to the full version of the paper [3]. Here, we also describe and analyze an extractor for the CBC mode.

In the following subsections we discuss ways of achieving PA1 assuming a nonce and arbitrary IV. Our basic building block will be a random IV PA1 scheme.

5.3 PA1 Nonce IV Schemes

Nonce IV schemes are not necessarily PA1 in general. For example, CTR mode with a nonce IV is not PA1 and in [3] we show that IND-CPA is distinct from PA1. Furthermore, coming up with a generic technique which transforms nonce IV schemes into PA1 schemes in an efficient manner is most likely not possible.

If we assume that the nonce IV scheme, when used as a random IV scheme, is PA1, then there is an efficient way of making the nonce IV scheme PA1. Note that we already have an example of a scheme satisfying our assumption: nonce IV CTR mode is not PA1, but RIV-CTR is.

Nonce Decoy. The *nonce decoy* method creates a random-looking IV from the nonce IV and forces the decryption algorithm to use the newly generated IV. Note that we are not only transforming the nonce into a random nonce: the solution depends entirely on the fact that the decryption algorithm does *not* recompute the newly generated IV from the nonce IV.

Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{V})$ be a nonce-IV-based AE scheme. For simplicity assume $\mathsf{IV} := \{0,1\}^n$, so that IVs are of a fixed length n. We prepare a pseudo-random function $G_{K'} : \mathsf{IV} \to \mathsf{IV}$ with an independent key K'. We then construct an AE scheme $\widetilde{\Pi} = (\widetilde{\mathcal{E}}, \widetilde{\mathcal{D}}, \widetilde{\mathcal{V}})$ as follows.

$\underline{\widetilde{\mathcal{E}}^{IV,A}_{K,K'}(M)}$:	$\underline{\widetilde{\mathcal{D}}^{IV,A}_{K,K'}(\widetilde{C}, T)}$:	$\underline{\widetilde{\mathcal{V}}^{IV,A}_{K,K'}(\widetilde{C}, T)}$:
$\widetilde{IV} \leftarrow G_{K'}(IV)$	$\widetilde{IV} \| C \leftarrow \widetilde{C}$	$\widetilde{IV}^* \leftarrow G_{K'}(IV)$
$(C, T) \leftarrow \mathcal{E}_K^{\widetilde{IV},A}(M)$	$M \leftarrow \mathcal{D}_K^{\widetilde{IV},A}(C, T)$	$\widetilde{IV} \| C \leftarrow \widetilde{C}$
$\widetilde{C} \leftarrow \widetilde{IV} \| C$	**return** M	$b \leftarrow \mathcal{V}_K^{\widetilde{IV},A}(C, T)$
return $(\widetilde{C}, T)$		**return** $(\widetilde{IV}^* = \widetilde{IV}$ and $b = \top)?\top : \bot$

Note that the decryption algorithm $\widetilde{\mathcal{D}}$ does not make use of K' or IV. If the decryption algorithm recomputes $\widetilde{IV}$ using K' and IV, then $\widetilde{\Pi}$ will not be PA1. Furthermore, one can combine $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{V}}$ in order to create a scheme which rejects ciphertexts when the IV it receives does not come from an encryption query.

The condition that Π with random IVs be PA1 is necessary and sufficient in order for $\widetilde{\Pi}$ to be PA1, assuming G is a PRF; see [3] for the proof of this statement. In Sect. 6.2 we discuss what the nonce decoy does for INT-RUP.

5.4 PA1 Arbitrary IV Schemes

PRF-to-IV. Using a technique similar to MAC-then-Encrypt [9], we can turn a random IV PA1 scheme into an arbitrary IV PA1 scheme.

The idea behind the *PRF-to-IV* method is to evaluate a VIL PRF over the input to the scheme and then to use the resulting output as an IV for the random IV encryption scheme. Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{V})$ be a random IV PA1 scheme taking IVs from $\{0,1\}^n$, and let $G : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ be a VIL PRF.

$\underline{\widetilde{\mathcal{E}}^{IV,A}_{K,K'}(M)}$:
$\widetilde{IV} \leftarrow G_{K'}(IV \| A \| M)$
$(C,T) \leftarrow \mathcal{E}^{\widetilde{IV},A}_{K}(M)$
return $(C, \widetilde{IV} \| T)$

$\underline{\widetilde{\mathcal{D}}^{IV,A}_{K,K'}(C, \widetilde{IV} \| T)}$:
$M \leftarrow \mathcal{D}^{\widetilde{IV},A}_{K}(C,T)$
return M

$\underline{\widetilde{\mathcal{V}}^{IV,A}_{K,K'}(C, \widetilde{IV} \| T)}$:
$M \leftarrow \widetilde{\mathcal{D}}^{IV,A}_{K,K'}(C, \widetilde{IV} \| T)$
$IV^* \leftarrow G_{K'}(IV \| A \| M)$
$b \leftarrow \mathcal{V}^{\widetilde{IV},A}_{K}(C,T)$
return $(\widetilde{IV} = IV^* \text{ and } b = \top)? \top : \bot$

The PRF-to-IV method is more robust than the nonce decoy since $\widetilde{\mathcal{D}}$ really only can use $\widetilde{IV}$ to decrypt properly.

The condition that Π with random IVs be PA1 is necessary and sufficient in order for $\widetilde{\Pi}$ to be PA1, assuming G is a VIL-PRF; see [3] for the proof of this statement. Note that the PRF-to-IV method is the basic structure behind SIV, BTM, and HBS. We show that the PRF-to-IV method is INT-RUP in Sect.6.2.

Online Encryption. Since the PRF needs to be computed over the entire message before the message is encrypted again, the PRF-to-IV method does not allow for online encryption. Recall that an encryption scheme has online encryption if for all (K, IV, A), the resulting function is online. Examples of such schemes include COPA and McOE-G.

If we want encryption and decryption to both be online in the arbitrary IV setting, then a large amount of ciphertext expansion is necessary, otherwise a distinguisher similar to the one used in the proof of Prop. 1 can be created.

An encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ is online if for some n there exist functions f_i and f'_i such that

$$\mathcal{E}_K(M) = f_n(M_1)\, f_{2n}(M_1 M_2) \cdots f_{jn}(M_1 M_2 \cdots M_j)\, f'_{|M|}(M) \ ,$$

where $j = \lfloor (|M| - 1)/n \rfloor$ and M_i is the ith n-bit block of M. If the encryption scheme has online decryption as well, then the decryption algorithm can start decrypting each "block" of ciphertext, or

$$\mathcal{D}_K(f_n(M_1)\, f_{2n}(M_1 M_2) \cdots f_{in}(M_1 M_2 \cdots M_i)) = M_1 M_2 \cdots M_i \ ,$$

for all $i \leq j$.

Proposition 3. *Let $\Pi = (\mathcal{E}, \mathcal{D})$ be an encryption scheme where $\mathcal{E}$ is n-online for all K, IV, and A, and $\mathcal{D}$ is online as well, then there exists a PA1-adversary $\mathbf{D}$ such that for all extractors $\mathbf{E}$ there exists an IND-CPA adversary $\mathbf{D}_1$ such that*

$$1 - \mathsf{CPA}_{\Pi}(\mathbf{D}_1) \leq \mathsf{PA1}^{\mathbf{E}}_{\Pi}(\mathbf{D}) \ ,$$

where $\mathbf{D}$ makes one $\mathcal{O}_1$ query, one $\mathcal{O}_2$ query, and $\mathbf{D}_1$ is as efficient as $\mathbf{D}$ plus one query to $\mathbf{E}$.

Proof. See [3]. □

Example 2. In certain scenarios, padding the plaintext is sufficient for PA1. Doing so makes schemes such as McOE-G secure in the sense of PA1, while keeping encryption and decryption online. The cost is a substantial expansion of the ciphertext. For the case of McOE-G, the length of the ciphertext becomes roughly twice the size of its plaintext.

It is important to note that McOE-G is based on an n-bit block cipher, and each n-bit message block is encrypted (after it is XORed with some state values) via the block cipher call. Since the underlying block cipher is assumed to be a strong pseudo-random function (SPRP), we can pad a message $M = M_1 M_2 \cdots M_\ell$ (each M_i is an $n/2$-bit string) as $0^{n/2} M_1 \parallel 0^{n/2} M_2 \parallel \cdots \parallel 0^{n/2} M_\ell$ and then encrypt this padded message using McOE-G. So each block cipher call processes $0^{n/2} M_i$ for some i. This "encode-then-encipher" scheme [12] is PA1 as shown in [3].

Example 3. If we do not require the decryption to be online, then we can achieve PA1 without significant ciphertext expansion. An example of a scheme that falls into this category is the recently-introduced APE mode [2], whose decryption is backward (and hence not online). A proof of this is given in [3].

5.5 PA2 Schemes

Most AE schemes are proven to be IND-CPA and INT-CTXT, which allows one to achieve IND-CCA [9] assuming verification works correctly. In order to be as efficient as possible, the underlying encryption schemes in the AE schemes are designed to only achieve IND-CPA and not IND-CCA, since achieving IND-CCA for encryption usually requires significantly more operations. For example, GCM, SIV, BTM, and HBS all use CTR mode for encryption, yet CTR mode is not IND-CCA. Since IND-CPA+PA2 is equivalent to IND-CCA′, none of these schemes achieve PA2.

A scheme such as APE also cannot achieve IND-CCA′ because its decryption is online "in reverse". If $(\mathcal{E}_K, \mathcal{D}_K)$ denotes APE, then an adversary can query $\mathcal{E}_K(M_1 M_2) = C_1 C_2$ and then $\mathcal{D}_K(C_1' C_2)$, which equals $M_1' M_2$. But if an adversary interacts with $(\$, \mathcal{D}_K)$ (see Def. 3), then $\mathcal{D}_K(C_1' C_2)$ will most likely not output $M_1' M_2$.

Existing designs which do achieve PA2 include those which are designed to be IND-CCA′, such as the solutions presented by Bellare and Rogaway [12], Desai [19], and Shrimpton and Terashima [36]. These solutions cannot be online, and they are usually at least "two-pass", meaning the input is processed at least twice.

6 Integrity in the INT-RUP Setting

6.1 INT-RUP Attack

Several AE schemes become insecure if unverified plaintext is released. In Proposition 4, we explain that OCB [33] and COPA [4] are not secure in the RUP setting.

The strategy of our attack is similar to that of Bellare and Micciancio on the XHASH hash function [8]. However, our attack is an improved version that solves a system of linear equations in $GF(2)$ with only half the number of equations and variables.

The attack works by first querying the encryption oracle under nonce N to get a valid ciphertext and tag pair. Then, two decryption queries are made under the same nonce N. Using the resulting plaintexts a system of linear equations is set up, which when solved will give the a forgery with high probability. A formal description of the attack is given in [3].

Proposition 4. *For OCB and COPA, for all $\ell \geq n$ there exists an adversary* $\mathbf{A}$ *such that*

$$\mathsf{INT\text{-}RUP}_{\Pi}(\mathbf{A}) \geq 1 - 2^{n-\ell} \ ,$$

where $\mathbf{A}$ *makes one encryption query and two decryption queries, each consisting of ℓ blocks of n bits. Then, the adversary solves a system of linear equations in $GF(2)$ with n equations and ℓ unknowns.*

6.2 Nonce Decoy and PRF-to-IV

In Sect. 5 we introduced a way of turning a random IV PA1 scheme into a nonce IV PA1 scheme, the nonce decoy, and a way of turning a random IV PA1 scheme into an arbitrary IV PA1 scheme, the PRF-to-IV method. Here we consider what happens to INT-RUP when the two methods are applied.

The nonce decoy adds some integrity to the underlying random IV PA1 scheme. Using the notation from Sect. 5.3, Π needs to be a slightly lighter form of INT-RUP in order for $\widetilde{\Pi}$ to be INT-RUP. Concretely, Π only needs to be INT-RUP against adversaries which use IVs which are the result of an encryption query. Furthermore, this requirement on Π is sufficient to prove that $\widetilde{\Pi}$ is INT-RUP.

Naturally if Π is INT-RUP, then $\widetilde{\Pi}$ is INT-RUP as well. In fact, if Π is INT-RUP against adversaries which use IVs which are the result of an encryption query, then $\widetilde{\Pi}$ is INT-RUP. These statements and their proofs can be found in [3].

The PRF-to-IV method is a much stronger transform than the nonce decoy. Following the notation from Sect. 5.4, we do not need to assume anything about the underlying random IV scheme Π in order to prove that $\widetilde{\Pi}$ is INT-RUP.

7 Conclusions

Many practical applications desire that an AE scheme can securely output plaintext before verification. We formalized security under the release of unverified plaintext (RUP) to adversaries by separating decryption and verification.

Two symmetric-key notions of plaintext awareness (PA1 and PA2) were introduced. In the RUP setting, privacy is achieved as a combination of IND-CPA and PA1 or PA2. For integrity, we introduced the INT-RUP notion as an extension of INT-CTXT, where a forger may abuse unverified plaintext. We connected our notions of privacy and integrity in the RUP setting to existing security notions, and saw that the relations and separations depended on the IV type.

The CTR and CBC modes with a random IV achieve IND-CPA+PA1, but this is non-trivial for nonce-based or deterministic encryption schemes. Our results showed that many AE schemes such as GCM, CCM, COPA, and McOE-G are not secure in the RUP setting. We provided remedies for both nonce-based and deterministic AE schemes. For the former case, we introduced the *nonce decoy* technique, which allowed to transform a nonce to a random-looking IV. The PRF-to-IV method converts random IV PA1 schemes into arbitrary IV PA1 schemes. We showed that deterministic AE schemes cannot be PA1, unless the decryption is offline (as in APE) or there is significant ciphertext expansion.

Future Work. Given that our PRF-to-IV method is rather inefficient, we leave it as an open problem to efficiently modify any encryption-only scheme into an AE scheme that is INT-RUP. A related problem is to fix OCB and COPA to be INT-RUP in an efficient way. The PA1 solutions we provide all start with the assumption that the nonce IV or arbitrary IV scheme is PA1 when a random IV is used instead. An interesting problem is to find alternative solutions to constructing nonce IV and arbitrary IV PA1 schemes. A problem of theoretical interest is to find a non-pathological random IV encryption scheme that is not PA1. In some applications, formalizing security in the RUP setting as IND-CPA+PA1 and INT-RUP may be sufficient. It is interesting to investigate how well this formalization reflects the problems encountered in real-world implementations, to see where PA2 may also be necessary, and how blockwise adaptive adversaries [25] play a role in the RUP setting. Finally, our paper does not address the behavior of applications which use unverified plaintext. A further understanding of the security risks involved in using unverified plaintext in applications is necessary.

Acknowledgments. The authors would like to thank Martijn Stam and the reviewers for their valuable comments.

References

1. AlFardan, N.J., Paterson, K.G.: Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In: IEEE Symposium on Security and Privacy. pp. 526–540. IEEE Computer Society (2013)
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: FSE. LNCS, Springer (2014)
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. Cryptology ePrint Archive, Report 2014/144 (2014), full version of this paper
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and Authenticated Online Ciphers. In: ASIACRYPT. LNCS, vol. 8269, pp. 424–443. Springer (2013)
5. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS. pp. 394–403. IEEE Computer Society (1997)
6. Bellare, M., Keelveedhi, S.: Authenticated and Misuse-Resistant Encryption of Key-Dependent Data. In: CRYPTO. LNCS, vol. 6841, pp. 610–629. Springer (2011)
7. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. ACM Tr. Inf. Sys. Sec. 7(2), 206–241 (2004)
8. Bellare, M., Micciancio, D.: A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost. In: EUROCRYPT. LNCS, vol. 1233, pp. 163–192. Springer (1997)
9. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: ASIACRYPT. LNCS, vol. 1976, pp. 531–545. Springer (2000)
10. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: ASIACRYPT. LNCS, vol. 3329, pp. 48–62. Springer (2004)
11. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: EUROCRYPT. LNCS, vol. 950, pp. 92–111. Springer (1994)
12. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: ASIACRYPT. LNCS, vol. 1976, pp. 317–330. Springer (2000)
13. Bellare, M., Rogaway, P.: Introduction to modern cryptography. In: UCSD CSE 207 Course Notes (September 2005)
14. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer (2012)
15. Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-Based Lightweight Authenticated Encryption. In: FSE. LNCS, vol. 8424, pp. 447–466. Springer (2013)
16. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: Security of Symmetric Encryption in the Presence of Ciphertext Fragmentation. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 682–699. Springer (2012)
17. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On Symmetric Encryption with Distinguishable Decryption Failures. In: FSE. LNCS, vol. 8424, pp. 367–390. Springer (2013)
18. Canvel, B., Hiltgen, A.P., Vaudenay, S., Vuagnoux, M.: Password Interception in a SSL/TLS Channel. In: CRYPTO. LNCS, vol. 2729, pp. 583–599. Springer (2003)

19. Desai, A.: New Paradigms for Constructing Symmetric Encryption Schemes Secure against Chosen-Ciphertext Attack. In: CRYPTO. LNCS, vol. 1880, pp. 394–412. Springer (2000)
20. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: FSE. LNCS, vol. 7549, pp. 196–215. Springer (2012)
21. Fouque, P.A., Joux, A., Martinet, G., Valette, F.: Authenticated On-Line Encryption. In: SAC. LNCS, vol. 3006, pp. 145–159. Springer (2003)
22. Goldwasser, S., Micali, S.: Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In: STOC 1982. pp. 365–377. ACM (1982)
23. Iwata, T., Yasuda, K.: BTM: A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption. In: SAC. LNCS, vol. 5867, pp. 313–330. Springer (2009)
24. Iwata, T., Yasuda, K.: HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption. In: FSE. LNCS, vol. 5665, pp. 394–415. Springer (2009)
25. Joux, A., Martinet, G., Valette, F.: Blockwise-Adaptive Attackers: Revisiting the (In)Security of Some Provably Secure Encryption Models: CBC, GEM, IACBC. In: CRYPTO. LNCS, vol. 2442, pp. 17–30. Springer (2002)
26. Katz, J., Yung, M.: Complete characterization of security notions for probabilistic private-key encryption. In: STOC. pp. 245–254. ACM (2000)
27. Krovetz, T., Rogaway, P.: The OCB Authenticated-Encryption Algorithm. `http://datatracker.ietf.org/doc/draft-irtf-cfrg-ocb` (June 2013)
28. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: INDOCRYPT. LNCS, vol. 3348, pp. 343–355. Springer (2004)
29. NIST: DES Modes of Operation. FIPS 81 (December 1980)
30. Paterson, K.G., AlFardan, N.J.: Plaintext-Recovery Attacks Against Datagram TLS. In: NDSS. The Internet Society (2012)
31. Rogaway, P.: Authenticated-encryption with associated-data. In: ACM Conference on Computer and Communications Security 2002. pp. 98–107. ACM (2002)
32. Rogaway, P.: Nonce-Based Symmetric Encryption. In: FSE 2004. LNCS, vol. 3017, pp. 348–359. Springer (2004)
33. Rogaway, P., Bellare, M., Black, J.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. ACM Tr. Inf. Sys. Sec. 6(3), 365–403 (2003)
34. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer (2006)
35. Rogaway, P., Wagner, D.: A Critique of CCM. Cryptology ePrint Archive, Report 2003/070 (2003)
36. Shrimpton, T., Terashima, R.S.: A modular framework for building variable-input-length tweakable ciphers. In: ASIACRYPT. LNCS, vol. 8269, pp. 405–423. Springer (2013)
37. Tsang, P.P., Smith, S.W.: Secure cryptographic precomputation with insecure memory. In: ISPEC 2008. LNCS, vol. 4991, pp. 146–160. Springer (2008)
38. Tsang, P.P., Solomakhin, R.V., Smith, S.W.: Authenticated streamwise on-line encryption. Dartmouth Computer Science Technical Report TR2009-640 (2009)
39. Vaudenay, S.: Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS. In: EUROCRYPT. LNCS, vol. 2332, pp. 534–546. Springer (2002)
40. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). Request For Comments 3610 (2003)
41. Wu, H., Preneel, B.: AEGIS: A Fast Authenticated Encryption Algorithm. In: SAC. LNCS, vol. 8282, pp. 185–201. Springer (2013)