Rogue Decryption Failures: Reconciling AE Robustness Notions

Guy Barwell, Daniel Page, and Martijn Stam

Department of Computer Science, University of Bristol,
Bristol, BS8 1UB, United Kingdom.
{guy.barwell, daniel.page, martijn.stam}@bristol.ac.uk

Abstract. An authenticated encryption scheme is deemed secure (AE) if ciphertexts both look like random bitstrings and are unforgeable. AE is a much stronger notion than the traditional IND–CCA. One shortcoming of AE as commonly understood is its idealized, all-or-nothing decryption: if decryption fails, it will always provide the *same single* error message *and nothing more*. Reality often turns out differently: encode-then-encipher schemes often output decrypted ciphertext before verification has taken place whereas pad-then-MAC-then-encrypt schemes are prone to distinguishable verification failures due to the subtle interaction between padding and the MAC-then-encrypt concept. Three recent papers provided what appeared independent and radically different definitions to model this type of decryption leakage.
We reconcile these three works by providing a reference model of security for authenticated encryption in the face of decryption leakage from invalid queries. Having tracked the development of AE security games, we provide a single expressive framework allowing us to compare and contrast the previous notions. We find that at their core, the notions are essentially equivalent, with their key differences stemming from definitional choices independent of the desire to capture real world behaviour.

Keywords: provable security, authenticated encryption, multiple errors, unverified plaintext, robustness

1 Introduction

Nowadays, authenticated encryption (AE) is understood to mean that ciphertexts both look like random bitstrings (IND$–CPA) and are unforgeable (INT–CTXT). Moreover, the customary syntax of AE considers encryption deterministic and stateless, instead accepting a nonce (number-used-once) and associated data to ensure that repeated encryption of the same message does not lead to repeated ciphertexts. Preferably security degrades gracefully if nonces are repeated. AE thus defined is more flexible and considerably stronger than the traditional notion of IND–CCA symmetric encryption.

The CAESAR competition [5] served as a catalyst to strengthen the security models used in AE even further. One particular shortcoming is the traditional reliance on an idealised, all-or-nothing decryption: if decryption fails, it

will only ever provide a single error message. For various reasons, this is not a realistic assumption. Especially MAC-then-encrypt schemes (or rather, decrypt-then-verify) are prone to real-world security flaws, on the one hand due to distinguishable verification failures and on the other due to the need to output (or at least, store) decrypted ciphertext before verification has taken place.

Three recent works improve the "robustness" of AE schemes by considering how well their security guarantees hold up under incorrect usage or when implemented non-ideally. Boldyreva et al. [7] investigated the effect of multiple decryption errors for both probabilistic and stateful encryption (BDPS). Later, Andreeva et al. [2] moved to a nonce-based setting, introducing a framework to capture the release of unverified plaintexts (RUP). Concurrently, Hoang et al. [12] coined an alternative notion, robust authenticated encryption (RAE), which they claim is radically different from RUP.

On the surface, these papers take very different approaches, with quite different goals in mind. BDPS concentrates on decryption errors, and does not consider nonce-based encryption. RUP extends AE by syntactically adding explicit, fixed-size tags and considering separate verification and decryption algorithms. It models the leakage of candidate plaintexts, with an eye on the online or nonce-abuse settings. In contrast, RAE considers schemes with variable, user-specified stretch as authentication mechanism, and decryption is given a much richer syntax, extending semantics for ciphertexts not generated by the encryption algorithm. This raises the natural questions how these models relate to each other and how well each captures real-world decryption leakage.

Our contribution. Inspired by the above works, we provide a framework taking in the best of all worlds, where our key goal is to reconcile RUP and RAE with BDPS, both notationally and conceptually. Our framework allows us to draw parallels and highlight where the works agree or differ, while ensuring any goals described can be easily interpreted and compared to the scenarios they model.

Our framework revolves around a broad reference game that models adversarial access, and demonstrate that classic reductions still hold. This allows us to define "subtle Authenticated Encryption" (SAE) as the strongest security goal relevant to (deterministic) decryption leakage. The term *subtle* highlights that security in the real-world is very much dependent of the subtleties of *how* decryption is implemented. As illustration, in the full version we describe a natural yet insecure implementation of AEZ [12], refuting its robustness. Finally, we compare results from the three noted papers within our framework, using SAE as a reference point. After clarifying some (misconceived) terminology, we find that for schemes *with fixed stretch* the notions essentially collapse.

The fundamental difference between the models is philosophical: Is authenticated encryption primarily a primitive like a blockcipher, whose security should be measured with reference to the ideal object of the given syntax and where the authentication level might be set to zero; or is it a means to authenticate and encrypt where security should be measured against a—possibly unobtainable—ideal?

2 Security Games for the Real World

2.1 Standard Syntax of Authenticated Encryption

Current understanding of authenticated encryption is the culmination of many years of work (see the full version [4] for an overview). Modern AEAD schemes take a number of standard inputs and produce a single output. The corresponding spaces are named after the elements they represent: the *key* space K, the *message* space M, the *nonce* space N, the *associated data* space A, and finally the *ciphertext* space C. Each of these spaces is a subset of $\{0,1\}^*$ and we make no assertions over the sizes of these spaces.

An *authenticated encryption* (AE) scheme is a pair of deterministic algorithms $\Pi = (\mathcal{E}, \mathcal{D})$ (encrypt and decrypt) satisfying

$$\begin{aligned} \mathcal{E} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \to \mathsf{C} \\ \mathcal{D} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \to \mathsf{M} \cup \{\perp\} \,. \end{aligned}$$

We use subscripts for keys, superscripts for public information (nonce and associated data) and put content data in parentheses.

To be *correct*, decryption must be a left inverse of encryption: if $C = \mathcal{E}_k^{N,A}(M)$ then $\mathcal{D}_k^{N,A}(C) = M$. Conversely, a scheme is *tidy* if decryption is a right inverse: if $\mathcal{D}_k^{N,A}(C) = M \neq \perp$ then $\mathcal{E}_k^{N,A}(M) = C$. Together then, correctness and tidiness imply encryption and decryption are inverses. For schemes that are both correct and tidy, $\mathcal{E}_k$ uniquely determines $\mathcal{D}_k$, which implies that security can be regarded as a property of $\mathcal{E}_k$ only [14].

The *stretch* measures the amount of ciphertext expansion (or redundancy). We require that the stretch $\tau(M) = |\mathcal{E}_k^{N,A}(M)| - |M|$, depends only on the length of the message, and so $\tau(M) = \tau(|M|)$ (for all k, N, A, and M). We call such schemes *τ-length-regular*, extending the accepted term *length–regular* to describe *how* the length is regulated. To minimise ciphertext expansion, most modern schemes set τ to be constant. We restrict ourselves to length-regular schemes: those whose stretch depends only on the length of the message, meaning $\tau(M) = \tau(|M|)$.

One might deviate from the syntax above. On the one hand, RUP uses an equivalent formulation with explicit tag space in addition to the ciphertext space (see Section 3.2). On the other hand, RAE uses an explicit input of the encryption indicating what size of tag is desired. In Section 3.3 we discuss the implication of user-defined tag-sized explicitly, and our rationale for omitting it from our framework.

2.2 Syntax of Subtle Authenticated Encryption

Just as a plan seldom survives contact with the enemy, so it goes with authenticated encryption: several provably secure schemes have fallen when implemented in practice. Especially for the decryption of invalid ciphertext it is challenging to ensure an adversary really only learns the invalidity of the ciphertext, and not

some additional information. Additional information that has been considered in the past (and we will encounter again shortly) are multiple error symbols and unverified plaintext. Both can be classified as *leakage*, leading to our new notion of a *subtle authenticated encryption* scheme.

A subtle AE (SAE) scheme is a triple of deterministic algorithms $\Pi = (\mathcal{E}, \mathcal{D}, \Lambda)$, where Λ corresponds to leakage from the decryption function. We restrict ourselves to leakage functions that are deterministic functions on their inputs, and only provide leakage to invalid decryption queries. Thus the leakage function looks like

$$\Lambda : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \to \{\top\} \cup \mathsf{L}$$

where the *leakage space* L can be any non-empty set not containing $\top$, and the distinguished symbol $\top$ refers to a message that is valid. So, for any (N, A, C), either $\mathcal{D}_k^{N,A}(C) = \bot$ or $\Lambda_k^{N,A}(C) = \top$, but not both: a message is either valid (and so decryption returns the plaintext but there is no leakage) or is invalid (and so decrypts to $\bot$ and leakage is available). The generality of L caters for any type of leakage, including schemes with multiple errors [7], those which output candidate plaintexts [2] or those which return arbitrary strings when presented invalid ciphertexts [12].

Explicitly separating Λ from $\mathcal{D}$ emphasises that leakage is a property of the decryption *implementation*, rather than of the decryption *function*. Consequently, security (for correct and tidy schemes) becomes a property of both the encryption function and the decryption implementation's leakage. A scheme may be proven secure for some leakage model Λ, but such a result is only meaningful as long as Λ accurately reflects the actual leakage as observed in practice. Even minor optimizations of the same decryption function can change the associated implementation so much that the scheme goes from being provably secure under some robust security definition to trivially insecure (we show how AEZ is affected in the full version). From this perspective, security becomes a subtle rather than robust affair, hence the name *subtle authenticated encryption*, a term inspired by the *SubtleCrypto* interface of the WebCryptoApi [18].

Comparison with the traditional model. There is a canonical mapping from any SAE scheme $(\mathcal{E}, \mathcal{D}, \Lambda)$ to a more traditional one $(\mathcal{E}, \mathcal{D})$ simply by removing access to the leakage oracle: correctness and tidiness of the subtle scheme clearly imply correctness and tidiness of the traditional one. Note that many distinct SAE schemes map to the same traditional form, implying that the canonical mapping induces an equivalence relation on SAE schemes. One could turn a traditional $(\mathcal{E}, \mathcal{D})$ scheme into a subtle form by inverting the above canonical map, for which the obvious preimage is setting $\Lambda^{N,A}(C) = \bot$ if $\mathcal{D}_k^{N,A}(C) = \bot$ and otherwise $\top$, again preserving correctness and tidiness. This corresponds to the SAE scheme whose implementation does not leak at all, so we expect our security notion to match the traditional one in this case (and it does).

Contrast with leakage resilience. Our separation into $\mathcal{D}$ and Λ is possible because decryption is deterministic, and its inputs (i.e. N, A, C) may be provided

to Λ_k. Within the leakage resilience community [11], leakage is generally characterised as an auxiliary output from the original algorithm (often supported by an auxiliary input to control the type of leakage); moreover one would expect both encryption and decryption to leak. This integrated perspective reflects the real world more closely (as leakage results from running some algorithm) and is more expressive. For example, if the decryption routine were probabilistic, the leakage may require access to the internal randomness, or if the scheme is stateful it may require the correct state variables. Some of these issues could be overcome by (for example) assuming the adversary always calls $\mathcal{D}_k$ directly before calling Λ_k, and that Λ_k has access to the previous internal state (from which it can deduce the operation of $\mathcal{D}_k$ if required), however ultimately which syntax works best depends on the context.

In the context of capturing subtle implementation differences for modern authenticated encryption (where decryption is stateless and deterministic) we feel separating leakage and decryption is a useful abstraction. Though our work could be recast into a form more closely aligned with the leakage resilience literature, the notation would become more cumbersome, for instance when an adversary can only observe the leakage.

2.3 Authentication and Encryption Security Games

In most modern AE definitions, an adversary is given access to a pair of oracles claiming to implement encryption and decryption. They are either real, and act as claimed, or ideal, returning the appropriate number of random bits for encryptions and rejecting all decryption attempts. To win the game, the adversary must decide which version it is interacting with. Certain queries would lead to trivial wins, for example asking for the decryption of a message output by the encryption oracle. These queries are forbidden (or their output suppressed).

This contrasts with the original definition of AE as IND–CPA plus INT–CTXT, where in both constituent games an adversary only has access to a single, real encryption oracle (and no decryption oracle); moreover, in the IND–CPA only a single challenge ciphertext is present and for INT–CTXT only a single ciphertext needs to be forged.

At first sight the two definitions may appear quite different, yet they are known to be equivalent. Where does this difference stem from and should one prefer one over the other?

We argue that both definitions can be cast as simplifications of a single reference game. This reference game is itself a distinguishing game where an adversary has access to two sets of oracles: one set of oracles will be used to capture the *goal* of the adversary, whereas the other matches the *powers* of the adversary. For instance, to capture AE an adversary has access to *four* oracles: the two oracles from the modern definition (implementing either the real or ideal scenario) *and* the two oracles from the traditional IND–CCA definition (namely true encryption and decryption oracles).

Four oracles may seem overly complicated, but we posit that our approach using a reference game has several advantages:

1. *Generality:* Hybrid arguments and composition results—the techniques implicitly underlying the standard definition—do not always hold when enriching the security model to take into account real-world phenomena such as key dependent messages or leakage (e.g. [10]). In these cases, one typically undoes certain simplifications; relying on our reference game instead is more transparent.
2. *Granularity:* Because adversarial goal and power are clearly separated, one can immediately identify a natural lattice of security notions and argue about possible equivalences *depending on the context*.
3. *Intuition:* The simplified games are less intuitive when considering real-life scenarios. For instance, even if an adversary knows it has seen a number of true plaintext–ciphertext pairs, for any set of fresh purported plaintext–ciphertext pair it should be clueless as to its validity. This statement follows directly from our reference game, yet for the simplified games one would need a hybrid argument.
4. *Tightness:* In real world scenarios, obtaining challenge ciphertexts versus known ciphertexts might carry different costs, which can be more easily reflected in our reference game (as the queries go to different oracles). A security analysis directly in our game is potentially more tight than one in a simplified game (whose results subsequently need to be ported to the more fine-grained real-world setting).

Security games. We refer to games in the form GOAL–POWER, clearly separating the adversary's objective from its resources. The complete lists of powers and goals are presented in Table 1, and described below. Security of scheme Π in game XXX against an adversary $\mathcal{A}$ is written as an advantage $\mathsf{Adv}_{\Pi}^{\mathrm{XXX}}[\Pi](\mathcal{A})$ and captures the adversary's ability to distinguish between two worlds. In both worlds the adversary has oracle access that depends on the scheme Π (initiated using some random and secret key $k \leftarrow\!\!\$\ \mathsf{K}$); the oracles corresponding to the goal differ between the worlds, whereas the oracles corresponding to the power will be identical. The notation $\Delta_{\mathcal{O}_a,\mathcal{O}_b}^{\mathcal{O}_1,\mathcal{O}_2}$, short-hand for the advantage in distinguishing between $(\mathcal{O}_1,\mathcal{O}_2)$ and $(\mathcal{O}_a,\mathcal{O}_b)$, is used to make the oracles explicit. A scheme is XXX *secure* if $\mathsf{Adv}_{\Pi}^{\mathrm{XXX}}$ is sufficiently small for all reasonably resourced adversaries.

Goals. The goal oracles Enc and Dec either implement the true scheme or an idealised version. In each case they return $\lightning$ if their inputs are not elements of the appropriate spaces. If $b = 0$, we are in the *real world*, where Enc and Dec implement $\mathcal{E}_k$ and $\mathcal{D}_k$ respectively, and if $b = 1$ we are in the *ideal world*, where they implement \$ and $\perp$.

The oracle $\perp$ matches the syntax of $\mathcal{D}_k$ but returns $\perp$ in response to any queries. The oracle \$ is a random function: for each nonce-associated data pair, it samples an element $\$_{N,A}$ uniformly at random from the set of all τ-length-regular functions $f : \mathsf{M} \to \mathsf{C}$. When queried, $\$(N, A, M) := \$_{N,A}(M)$. When

Oracles	Type / Role / Bit	Challenge Enc 1	Challenge Dec 2		Honest $\mathcal{E}_k$ 3	Honest $\mathcal{D}_k$ 4		Leakage Λ_k 5	
	Names	0	0	n/a	0	0	PAS	0	no leakage
		1	0	IND	1	0	CPA	1	leakage (s)
		0	1	CTI	0	1	CDA		
		1	1	AE	1	1	CCA		

Table 1: A compact table of goals and powers. The challenge oracles Enc and Dec specify the adversary's goal: they either implement honest encryption and decryption or their idealised versions, where Enc samples responses randomly and Dec returns $\perp$ to all queries. The honest oracles $\mathcal{E}_k$ and $\mathcal{D}_k$ capture the adversary's power. Each game corresponds to a 5–bit bitstring $b_1b_2b_3b_4b_5$, with for example CTI–CPA (equivalent to INT–CTXT) being 01100, and IND–sCCA (i.e. IND-CCA with decryption leakage) as 10111.

the adversary is forbidden from repeating queries, this corresponds to uniformly sampling $|M| + \tau(|M|)$ random bits.

The *goal* is defined based on which oracles an adversary is given access to. We code this access using 2-bit strings, where the first bit is set in the presence of an Enc oracle, leaving the second bit for Dec. This leads to three possible goals (it does not make sense to have no challenge oracle): indistinguishability (IND, 10), authenticated encryption (AE, 11), and ciphertext integrity (CTI, 01).

Ideal versus attainable. Our ideal encryption oracle responds random bitstrings for fresh calls. This corresponds to security as one would expect it to hold; it can be considered as a computational analogue of Shannon's notion of perfect security where the uncertainty of a ciphertext given a message should be maximal. Similarly, the ideal decryption oracle is unforgiven, implying (traditional) integrity of ciphertexts.

Consequently, for some classes of constructions the advantage cannot be small. For instance, for online schemes it will be easy to distinguish by looking at prefixed and for schemes without sufficient stretch, randomly choosing a ciphertext can be used to forge.

One could bypass these impossibilities by adapting the ideal oracles accordingly [1, 13]). Hoang et al. [12] suggest to use attainable security as benchmark; one can see the resulting security notions as (ever more complicated) extensions of the pseudorandom permutation notion typical for blockcipher security. This immediately reveals that to some extent, this choice is one of abstraction boundaries. When purely studying how to transform one primitive to another, it makes sense to used the ideal primitive as benchmark (as that will be the best attainable). Yet, we prefer a security definition that is both robust and meaningful: When non-experts use the primitive in larger designs, there should be as few implementation and configuration pitfalls as possible plus a small adversarial advantage should imply security as intuitively understood.

Powers. Traditionally, the adversary's *powers* describe what access they are given to honest encryption and decryption oracles, with which to learn about the scheme. Again, we identify these with 2-bit strings, listed in Table 1. The standard notions are a *passive* attack (PAS, 00) a *chosen plaintext* attack (CPA, 10), and a *chosen ciphertext* attack (CCA, 11). Access to only a decryption oracle is known as (DEM) CCA in the KEM–DEM setting (e.g. [8, 9]), we will refer to it as a *chosen decryption* attack (CDA, 01). Unless *overall* encryption access is restricted as in the DEM scenario, the CDA scenario is of limited relevance (see Section 2.5).

Leakage oracle. We add a third honest oracle implementing Λ_k, that models how schemes behave when subject to imperfect decryption implementations. Again, we use a bit to indicate whether a game provides an adversary access or not. If not, the standard notions arise, but presence leads to a range of new notions, which we will call their "subtle" variant. The name is chosen to emphasise that security critically depends on implementation subtleties.

As an example, power 101 stands for "subtle Chosen Plaintext Attack", or sCPA in short (note the "s" prefix). The power 001 corresponds to an adversary who cannot make decryption queries, yet it can observe leakage from them. This seeming contradiction makes sense when recalling that Λ_k only gives out information when queried with invalid ciphertexts. For instance, an adversary might learn how long it takes for ciphertexts to be rejected, but not what plaintexts correspond to valid ciphertexts. Given the implied validity checking capability and following the literature, we will refer to this power as a *chosen verification* attack (CVA) instead of a subtle passive attack.

2.4 Restrictions on the Adversary

With these lists in place, we consider what domain separations are required to prevent trivial wins. That is, we ask in what cases must the adversary be forbidden from taking the output of one oracle and using it as input to a second. The domain separation required for inputs to Λ_k is the same as $\mathcal{D}_k$, although we do not place any restrictions on the output of Λ_k: any seemingly trivial wins that occur from this are weaknesses of the scheme and demonstrate such a Λ_k cannot be secure. In the reference game, the adversary may make any queries he wishes that are not prohibited. In the effective game, he does not make superfluous queries either.

Trivial wins. Any messages repeated between the two encryption oracles will distinguish the Enc oracle. Similarly, attempting to decrypt the output of Enc will allow the adversary to immediate determine whether Enc is random, since he will receive the initial plaintext if not. Attempting to decrypt the output of the honest encryption oracle $\mathcal{E}_k$ will also trivially identify whether Dec is real or idealised. Since the scheme is assumed to be tidy, we have that for any $C \in \mathsf{C}$, $\mathcal{E}_k(\mathcal{D}_k(C)) = C$. So, any output from the honest decryption oracle

$\mathcal{D}_k$ cannot be passed to the challenge encryption oracle Enc, since this would trivially distinguish the schemes.

Superfluous queries. A superfluous query is one to which the adversary need not make. Sending the output of $\mathcal{E}_k$ to $\mathcal{D}_k$ is superfluous since by correctness the answer is already known. Similarly, tidiness implies the opposite: output from $\mathcal{D}_k$ need not be sent to $\mathcal{E}_k$. As soon as Dec outputs something other than $\perp$, the adversary can distinguish it as the real case, and so might as well terminate, meaning no outputs from Dec need ever be queried to the encryption oracles. Finally, though not displayed in the diagram, assuming the game is deterministic and stateless (such as in the nonce or IV–based settings) it is superfluous to repeat queries or make any that return $\lightning$, since neither yields useful information.

Nonces. If the adversary is *nonce-respecting* if he does not query (N, A, M') to either Enc or $\mathcal{E}_k$ if he has already queried either of them with (N, A, M) for some M. Note that we do not require the adversary be nonce-respecting, leaving this choice to specific security notion: relations between games are independent of strategies the adversary may or may not use, such as being nonce-respecting or nonce-abusing. That said, this behaviour can be enforced by the security game suppressing all such queries and returning $\lightning$, making such queries superfluous.

2.5 Effective Games

Since there are 32 possible games and countless probabilistic adversaries, it would be prudent to begin by removing those which are directly equivalent. We give these in terms of the corresponding bitstring, where `x`,`y` and `z` signify bits that may (but need not) be set. We write X $\Longrightarrow$ Y to signify that security in game X implies security in game Y, meaning that for any adversary $\mathcal{A}$ against game Y there is an adversary $\mathcal{B}$ against game X who uses similar resources and wins with similar probability.

Proposition 1 lists three (classes of) implications, which allows us to reduce the 32 games to only 4 interesting ones in Corollary 1. The proof for Proposition 1 can be found in the full version.

Proposition 1. *We may assume the adversary is deterministic and makes no superfluous or prohibited queries. Against such an adversary, several games are trivially related:*

1. *Adding extra oracles never makes the adversary weaker.*
2. ***x1y0z*** $\Longleftrightarrow$ ***x1y1z****: a decryption oracle does not help if a Dec challenge oracle is present.*
3. ***1x0yz*** $\Longleftrightarrow$ ***1x1yz****: an encryption oracle does not help if a Enc challenge oracle is present.*

However, no further (generic) reductions are possible.

Corollary 1. *The effective games are just 1100**x**, 1000**x**, 1001**x** and 0110**x** (where **x** signifies a bit that might or might not be set). These correspond to AE–PAS, IND–PAS, IND–CDA, CTI–CPA and their subtle variants.*

2.6 Error Simulatability

We now define ERR (for *Error Simulatability*) to be the goal of distinguishing Λ_k from Λ_l, where $l \leftarrow\!\!\$\ \mathsf{K}$ is drawn independently of k. As always, this can be paired with any set of powers, leading to (for example) ERR–CCA:

$$\mathsf{Adv}^{\mathrm{ERR-CCA}}_{\Pi} := \Delta^{\mathcal{E}_k,\mathcal{D}_k,\Lambda_k}_{\mathcal{E}_k,\mathcal{D}_k,\Lambda_l} \ .$$

Initially this may appear unnecessarily specific: why should a definition of simulatability be given that restricts the simulator so tightly? As the following lemma shows, if there exists any good simulator, then Λ_l is one. Choosing this as our reference definition means security is completely described by $(\mathcal{E}, \mathcal{D}, \Lambda)$, rather than also requiring a description of the simulator. Obviously proof authors are welcome to use any simulator they wish, but a reference definition should be no more complex than absolutely necessary. After providing the lemma in question, we give some initial observations. Both results are proven in the full version

Lemma 1. *If there exists a good simulator, Λ_l is one. That is, if there exists some stateful simulator S such that $\Delta^{\mathcal{E}_k,\mathcal{D}_k,\Lambda_k}_{\mathcal{E}_k,\mathcal{D}_k,\mathsf{S}}$ is small, then so is ERR–CCA. The inverse also holds.*

Lemma 2. *We observe that* $\mathsf{Adv}^{\mathrm{ERR-PAS}}_{\Pi} = 0$. *Also,* CTI–CPA + ERR–CCA $\iff$ CTI-sCPA + ERR–CPA

2.7 Subtle Authenticated Encryption (SAE)

We define *Subtle Authenticated Encryption* (SAE) as a more succinct name for AE-sCCA, the strongest goal describable within this framework (i.e. 11111). The name, inspired by WebCryptoAPI [18], highlights the importance of the subtleties in implementations when applying such results. Thus, a secure SAE scheme is a triple $(\mathcal{E}, \mathcal{D}, \Lambda)$ along with appropriate spaces such that the AE-sCCA advantage is sufficiently small. So, the adversary has access to challenge encryption and decryption oracles, as well as honest encryption and decryption oracles, and certain amounts of leakage from the decryption function, and can make any query that does not leak to a trivial win. This characterisation clearly describes the situation from the real-world perspective.

From the designers point of view, due to reductions described in Proposition 1 it suffices to demonstrate that the scheme is AE-sPAS (i.e. AE-CVA, 11001) against an adversary who does not make useless queries or those that lead to trivial wins. Clearly there are various ways of doing this. Looking ahead somewhat, we will provide description of SAE in terms of the RUP definitions, as well as comparing it with RAE. The most intuitive method for proving a scheme SAE secure is likely to be through the following decomposition.

Theorem 1. *The SAE goal can be trivially decomposed:*
SAE $\iff$ AE + ERR–CCA $\iff$ IND–CPA + CTI–CPA + ERR–CCA

Our notion	IND–CPA	CTI–CPA	CTI–sCPA	IND–sCCA	IND–CVA
Simplified bitstring	10000	01100	01101	10011	10001
BDPS notion	IND$-CPA	INT–CTXT*	INT–CTXT	IND$–CCA	IND$–CVA
Reference (in [7])	Def. 5	Def. 7	Def. 7	Def 5.	Def. 5
Direct translation	10000	01110	01111	10011	10001
RUP notion	IND–CPA	INT–CTXT	INT–RUP		
Reference (in [2])	Def 1.	Def. 4	Def. 8		
Direct translation	10000	01100	01111		

Table 2: Notions from BDPS and RUP that that directly translate into our framework.

3 Comparison of Recent AE Notions

Three recent papers introduced strengthened AE notions to capture distinguishable decryption failures [7], releasing unverified plaintext [2], and "robust" authenticated encryption [12]. In every case the encryption oracle can be cast as

$$\mathrm{E} \colon \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \to \mathsf{C}$$

but their authors make slightly different definitional choices depending on which aspect of the implementation they had in mind when developing the notion. The main differences are how decryption and its leakage are defined, when a ciphertext is considered valid, and what security to aim for. In the remainder of this section we will show how each of these three notions can be cast into our framework. With the appropriate modifications, it turns out that each of these three notions are essentially equivalent to our more general notion. As an obvious corollary, the three existing notions turn out to be not quite that radically different.

3.1 Distinguishable Decryption Failures (BDPS, [7])

Several provably secure IND-CCA secure schemes have succumbed to practical attacks as a result of different decryption failures being distinguishable, both in the public key and symmetric settings [6, 19]. Boldyreva et al. [7] initiated a systematic study of the effects of symmetric schemes with multiple decryption errors. They emphasised probabilistic and stateful schemes, omitting a more modern nonce-based treatment. Below we describe the nonce-based analogues of their syntax and security notions.

A nonce-based, multi-error AE scheme a la BDPS, is a pair $(\mathrm{E}_k, \mathrm{D}_k)$,

$$\begin{aligned} \mathrm{E} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{M} \to \mathsf{C} \\ \mathrm{D} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \to \mathsf{M} \cup \mathsf{L} \end{aligned}$$

satisfying the classical definition of correctness. The idea is that if decryption fails, it may output any error symbol from L. (BDPS stipulate finite L, but this restriction appears superfluous and we omit it.)

To cast a (nonce-based) BDPS scheme into our SAE syntax, we observe the obvious (invertible) mapping from a scheme $(\mathrm{E}_k, \mathrm{D}_k)$ by setting $\mathcal{E}_k = \mathrm{E}_k$, $\mathcal{D}_k(C) = \mathrm{D}_k(C)$ whenever $\mathrm{D}_k(C) \in \mathsf{M}$ or otherwise $\perp$, and $\Lambda_k(C) = \mathrm{D}_k(C)$ whenever $\mathrm{D}_k(C) \in \mathsf{L}$ or otherwise $\top$.

Notions. BDPS define a number of notions, including both IND and IND$ concepts. Once adapted to a nonce-based setting, several of their notions directly translate into our framework, as listed in Table 2. Additionally, BDPS define *error invariance* [7, Def. 8], which (roughly) says that it should be hard for an adversary with access to honest encryption and decryption oracles to achieve any leakage other than a particular value. This notion, INV–ERR, implies an adversary cannot learn anything from decryption leakage and can be thought of as a special case of ERR–CCA since the simulator need just return this common value. However, error invariance is strictly stronger than leakage simulatability.

The strongest goal defined by BDPS is IND$–CCA3 [7, Def. 19], which incorporates multiple errors to the classical authenticated encryption notion. It is characterised by two oracles: an adversary has to distinguish between $(\mathrm{E}_k, \mathrm{D}_k)$ (real) and $(\$, \perp)$ (ideal), where the error $\perp$ is a parameter of the notion. Thus despite the desire to capture multiple errors, in the ideal case the adversary is still only presented with a single error symbol. This curious artefact results from using INV–ERR rather than ERR–CCA to characterise "acceptable" leakage. Unfortunately, it leads to a reference definition that does not model the real-world problem satisfactorily, for instance it fails to capture the release of unverified plaintext.

Implications and separations. BDPS provide several implications and separations between their notions. Although originally stated and proven for probabilistic and stateful schemes, the results easily carry over to a nonce-based setting. Using our naming convention, BDPS show that IND–CVA + CTI–sCPA $\implies$ IND–sCCA, yet IND–CVA + CTI–CPA $\not\Longrightarrow$ IND–sCCA. This immediately implies a separation between CTI–sCPA and CTI–CPA. They also prove that AE and INV–ERR jointly are equivalent to their IND$–CCA3 notion (Thm. 20), which itself implies IND–CVA and CTI–sCPA.

Since INV-ERR implies ERR–CCA, this means IND$–CCA3 implies SAE. Moreover, the separation between ERR–CCA and INV–ERR carries over when comparing IND$–CCA3 and SAE. For completeness, we give the following theorem, which is a direct result of combination of Theorem 1 and Theorem 20 of BDPS (after incorporating nonces) with the observation that INV–ERR is more restrictive than ERR–CCA.

Theorem 2. *The IND$–CCA3 notion of BDPS is stronger than SAE solely in its requirement of simulatable errors. That is,*

IND$–CCA3 $\iff$ AE + INV–ERR $\implies$ AE + ERR–CCA $\iff$ SAE

3.2 Releasing Unverified Plaintext (RUP, [2])

Andreeva et al. [2] set out to model decryption more accurately for schemes that calculate a candidate plaintext before confirming its validity. In practice, such a candidate plaintext often becomes available (including to an adversary), even if validation fails. Examples include all schemes that need to decrypt or decipher before integrity can be checked (covering MAC–then–Encrypt, MAC–and–Encrypt, and encode–then–encipher) as well as schemes sporting online decryption (for instance single-pass CBC–then–MAC decryption). Andreeva et al. provide a large number of new definitions, covering security under decryption-leakage for both confidentiality and integrity.

Differences between frameworks. The RUP framework includes an explicit tag T, however the tag and ciphertext terms are always used together. This allows us to consider the ciphertext as (C, T) instead, which can be injectively mapped into C, e.g. by $C' := C||T$ if the stretch is fixed. Following their motivating scenario, the RUP paper models decryption using a decryption oracle D and a verification oracle V satisfying

$$\mathrm{D} : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \to \mathsf{M} = \mathsf{L}$$
$$\mathrm{V} : \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathsf{C} \to \{\top, \bot\} \ .$$

When called with a valid ciphertext, D_k returns the plaintext, and V_k returns $\top$. Conversely, when called with an invalid ciphertext, D_k will return some leakage information (nominally, the eponymous "unverified plaintext") and V_k will return $\bot$.

By changing perspective, we can cast a RUP scheme into the SAE framework: let $\mathcal{D}_k(C) = \mathrm{D}_k(C)$ if $\mathrm{V}_k(C) = \top$ (otherwise $\mathcal{D}_k(C) = \bot$) and $\Lambda_k(C) = \mathrm{D}_k(C)$ whenever $\mathrm{V}_k(C) = \bot$ (and otherwise $\Lambda_k(C) = \top$). Then, $(\mathcal{E}, \mathcal{D}, \Lambda)$ is an SAE scheme (where $\mathcal{E} = \mathrm{E}$), with leakage space $\mathsf{L} = \mathsf{M}$.

Notions. Andreeva et al. refer to the classic "encryption-only" notions of confidentiality and integrity under their customary names IND–CPA and INT–CTXT (our CTI–CPA). When decryption comes into play, a large number of new notions is suggested, typically defined in terms of adversarial access to their D_k and V_k oracles.

For integrity, dubbed INT–RUP for integrity under release of unverified plaintext, the adversary is given full access to all three honest oracles ($\mathcal{E}_k, \mathrm{D}_k$, and V_k), and challenged to make a forgery. INT–RUP directly translates into our framework, where it corresponds to CTI-sCCA (itself equivalent to CTI–sCPA). This makes Andreeva et al.'s INT–RUP equivalent to BDPS's INT–CTXT notion.

For the myriad of RUP's confidentiality notions, an adversary is—for whatever reason—not provided with a verification oracle. This makes translation into our syntax cumbersome as any direct method would implicitly provide access to V_k functionality.

Implications and separations. Andreeva et al. provide a number of implications and separations involving their new notions. They show that PA2 and DI are equivalent (Thms. 8,9), and imply PA1 (Thm. 1). Moreover, when combined with IND–CPA, PA2 provides a meaningful increase in security (Thms. 2,3), whereas PA1 does not (Thms. 4,5). Finally, they provide an alternative proof that CTI–sCPA is strictly stronger than CTI–CPA (Thm. 10).

Comments and comparisons. The RUP model restricts any decryption leakage to the message space. This is unnecessarily restrictive: it does not directly cover multiple decryption errors; moreover a scheme may conceivably leak some internal variable (say a buffer) that is not in the message space.

The verification oracle for most of the RUP confidentiality notions is missing. For instance, the RUP version of IND–CCA security only gives an adversary access to the leakage, which raises the question whether RUP's IND–CCA security implies classical IND–CCA once the leakage is ignored. If the scheme is tidy, the RUP decryption and encryption oracle together suffice to implement the verification oracle. For a tuple (N, A, C), request $M \leftarrow \mathrm{D}_k^{N,A}(C)$ and "accept" if and only if $C = C' \leftarrow \mathrm{E}_k^{N,A}(M)$. Unfortunately, the domain separation in place for RUP's IND–CCA prohibits this sequence of queries. As a result, it is unclear whether RUP's IND–CCA implies standard IND–CCA or not, even though the former is defined as part of a framework of stronger notions.

Authenticated Encryption definition. Andreeva et al. suggest that an authenticated encryption should meet the combined goals of IND-CPA and PA for confidentiality, and INT-RUP for integrity [2, §8]. Having to safisfy three separate notions may appear needlessly complicated and lacks the elegance a single notion can provide. We propose *RUPAE* as a natural and neater way of defining Andreeva et al.'s final objective, where we use DI instead of the less direct PA:

$$\mathsf{Adv}^{\mathrm{RUPAE}}_{\Pi} := \Delta^{\mathrm{E}_k,\mathrm{D}_k,\mathrm{V}_k}_{\$\ ,\mathrm{D}_l\ ,\ \perp}$$

This goal may originally have been envisaged by the authors, yet it was not explicitly alluded to (let alone defined). Providing a single succinct security goal is only worthwhile if it properly captures the the compound notions, which we show in Thm. 3. The proof is intuitive, based around liberal use of the triangle inequality, see the full version for details.

Theorem 3. *The single term RUPAE notion is equivalent to the triple of goals originally proposed. That is,*

RUPAE $\iff$ CTI–sCPA + DI + IND–CPA $\iff$ INT–RUP + PA + IND–CPA

To relate this to our other notions, we provide the following observation(proven in the full version):

Lemma 3. CTI–sCPA + ERR–CPA $\iff$ CTI–sCPA + DI

Finally then, we have the reassuring result that security within the RUP framework coincides with our more general definition. To prove it, one simply chains Theorem 1, Lemmas 2 and 3, then Theorem 3 (in that order).

Corollary 2. *RUPAE security is equivalent to SAE security.*

3.3 Robust Authenticated Encryption (RAE, [12])

Robust authenticated encryption, as proposed by Hoang et al. [12, §3], has robustness against inadvertent leakage of unverified plaintext as one of it goals. A notable difference between traditional notions of AE and RAE is that the latter explicitly targets schemes where the intended level of integrity is specified by the user for each message. To this end, both encryption and decryption algorithms are provided with an additional input, called the stretch parameter τ, leading to the syntax:

$$\begin{aligned} \mathrm{E} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathbb{N} \times \mathsf{M} \to \mathsf{C} \\ \mathrm{D} &: \mathsf{K} \times \mathsf{N} \times \mathsf{A} \times \mathbb{N} \times \mathsf{C} \to \mathsf{M} \,. \end{aligned}$$

Thus encryption calls are of the form $C = \mathrm{E}_k(N, A, \tau, M)$, taking in a nonce N, some associated data A, the stretch parameter τ and a message M, and output some ciphertext C. There is a requirement that τ is indeed the stretch, namely that $|C| = |M| + \tau$. Decryption calls take a similar format, and are allowed to "leak" a string *not* of the correct length when queried with invalid inputs. This length restriction on the leakage implies that valid ciphertexts can easily be determined from their length: if $M = \mathrm{D}_k(N, A, \tau, C)$ and $|M| = |C| - \tau$, then it follows that $\mathrm{E}_k(N, A, \tau, M) = C$.

The security game. The RAE security game aims to describe the *best possible* security for an object with the given syntax. Comparison to ideal objects is not new: it is the standard notion for blockciphers (namely a strong pseudorandom permutation) and has appeared previously as an alternative for deterministic authenticated encryption (namely strong pseudorandom injections).

For given stretch τ, the ideal object is a random element of $\mathrm{Inj}(\tau)$, the set of all injective functions whose outputs are always τ bits longer than their input. The inverse of an element $\pi \in \mathrm{Inj}(\tau)$ is not well defined (for $\tau > 0$) for strings outside of the range π. Since decryption may leak on these incorrect ciphertexts, returning $\perp$ in that case is no longer an option. Hoang et al. solve this problem by introducing a simulator S_π which has very restricted "access" to the ideal encryption π, as explained below.

Security is then defined relative to a simulator S and in terms of distinguishing between two worlds, with

$$\mathsf{Adv}^{\mathrm{RAE}}_{\Pi,\mathsf{S}} := \mathbb{P}\left[k \leftarrow\!\!\$\, \mathsf{K} \colon \mathcal{A}^{\mathrm{E}_k, \mathrm{D}_k} \to 1\right] - \mathbb{P}\left[\pi_{N,A,\tau} \leftarrow\!\!\$\, \mathrm{Inj}(\tau) \colon \mathcal{A}^{\pi, \mathsf{S}_\pi} \to 1\right] \,.$$

Here the injections $\pi_{N,A,\tau}$ are tweaked by the nonce, associated data, and stretch τ. Decryption queries in the ideal world are answered by S_π which exhibits the following behaviour. If a decryption query is valid, then it is of the form

(N, A, τ, C) where $C \in \mathsf{Image}(\pi_{N,A,\tau})$ and the simulator S_π returns the preimage M. Otherwise, the ciphertext is invalid, or $C \notin \mathsf{Image}(\pi_{N,A,\tau})$. In this case, the oracle calls a stateful simulator S, which must simulate the decryption oracle and output a bitstring of any length other than $|C| - \tau$, *without* access to the injections $\pi_{\cdot,\cdot,\cdot}$ (and the S_π oracle will simply forward S's output). A code-based description of this can be found in the original paper, where it is referred to as world $\mathbf{RAE}_{\Pi,S}$ [12, Fig. 2].

Fixing the stretch. The variable, user-defined stretch sets RAE apart from the notions discussed in this paper so far. Although Hoang et al. insist that all values of stretch should be allowed for a scheme to be RAE, including $\tau = 0$, they hasten to add that this does make forging trivial, making it impossible to get a good (generic) upper bound on the CTI-CPA advantage. However, there is no intrinsic reason not to let a scheme restrict which values of τ it supports. Certainly their security definition still makes perfect sense if the stretch is no longer user defined and depends only on the length of the input message.

To ease comparison with previous security notions, we will henceforth restrict attention to fixed stretch schemes. This makes the mapping that takes an RAE scheme to an SAE scheme rather intuitive, and analogous to that used in Section 3.2. Explicitly, let (E, D) be an RAE scheme, and (inspired by RUP) let V_k be the associated validity function, where $\mathrm{V}_k^{N,A,\tau}(C) = \top \iff |\mathrm{D}_k^{N,A,\tau}(C)| - |C| = \tau$. Then $(\mathcal{E}, \mathcal{D}, \Lambda)$ is an SAE scheme, where $\mathcal{E}_k^{N,A}(M) := \mathrm{E}_k^{N,A,\tau}(M)$ and

$$\mathcal{D}_k^{N,A}(C) := \begin{cases} \mathrm{D}_k^{N,A}(C) \\ \bot \end{cases} \quad , \qquad \Lambda_k^{N,A}(C) := \begin{cases} \top & \text{if } \mathrm{V}_k^{N,A,\tau}(C) = \top \\ \mathrm{D}_k^{N,A}(C) & \text{if } \mathrm{V}_k^{N,A,\tau}(C) \neq \top \end{cases}$$

Clearly this security game is similar to those presented above.

Comments and comparisons. Following Rogaway's definitional papers [15–17], most recent symmetric results have been given in terms of indistinguishability from the ideal world $(\$, \bot)$: an ideal encryption oracle that outputs random bits and an ideal decryption oracle that never accepts. Hoang et al. instead opt for an ideal world that corresponds to the "best achievable", a contrast they emphasize: "Before, AE–quality was always measured with respect to an aspirational goal; now we're suggesting to employ an achievable one." [12, §1:Discussion].

One feature, possibly by design, of RAE is that it accurately describes the security attainable from a PRP through the encode-then-encipher paradigm. Leakage is envisaged as being an invalid final buffer: one that has been deciphered but did not decode. This leads to the slightly artificial restriction that leakage cannot be a string of valid length.

Fixed–stretch RAE as an SAE goal. Having applied the transform (which has no bearing on security), it is not surprising to find RAE and SAE security

essentially coincides, with the only complication a generic attacks term, reflecting the difference between ideal and best possible security. After providing the RAE[τ] analogue of Lemma 1, we provide an explicit relationship between the games. (Neither proof is complicated, and both can be found in the full version.)

Lemma 4. *For any simulator* S, $\mathsf{Adv}^{\mathrm{RAE}[\tau]}_{\Pi,\Lambda}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathrm{RAE}[\tau]}_{\Pi,\mathsf{S}}(\mathcal{A})$, *where* Λ *is to the simulator that first samples* $l \leftarrow\!\!\$\ \mathsf{K}$, *then for all queries evaluates* Λ_l.

Theorem 4. *RAE[τ] and SAE security are equivalent. Explicitly, for an adversary* $\mathcal{A}$ *making at most* q *queries, and using a repeated nonces* r *times,*

$$\left|\mathsf{Adv}^{\mathrm{RAE}[\tau]}_{\Pi,\Lambda}(\mathcal{A}) - \mathsf{Adv}^{\mathrm{SAE}}_{\Pi}(\mathcal{A})\right| \leq \tfrac{q}{2^{\tau-1}} + \tfrac{r^2+r}{2^{\tau+m+1}} \ .$$

4 Conclusions

By defining SAE we provided a framework useful to compare prior notions all addressing the same problems, but from slightly differing perspectives. BDPS provides the most generalised syntax, although a (seemingly unnecessary) condition that the error space be finite limits the applicability of their results. RUP presents the material in a very practical way, with definitions and models that clearly describe how decrypt-then-verify schemes behave, but in doing so yield a scheme that does not readily generalise to handling alternative leakage sources. RAE on the other hand defines a goal that, at first glance, appears to be the strongest of them all, but upon further inspection is rather more nuanced. Overall, the three recent works have more in common than the original authors (esp. of RUP and RAE) might have indicated.

Acknowledgements. We thank Dan Martin and Elisabeth Oswald for fruitful discussions regarding leakage-resilience and the anonymous referees of the IMA International Conference on Cryptography and Coding 2015 for their constructive feedback.

This work was conducted whilst Guy Barwell was a PhD student at the University of Bristol, supported by an EPSRC grant.

References

1. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: Don't Panic! The Cryptographers' Guide to Robust Authenticated (On-line) Encryption. Comments to CAESAR mailing list (2015)
2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 105–125. Springer, Heidelberg (Dec 2014)
3. Atluri, V. (ed.): ACM CCS 02. ACM Press (Nov 2002)
4. Barwell, G., Page, D., Stam, M.: Rogue decryption failures: Reconciling ae robustness notions

5. Bernstein, D.J.: CAESAR competition call (2013), http://competitions.cr.yp.to/caesar-call-3.html
6. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (Aug 1998)
7. 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)
8. Davies, G.T., Stam, M.: KDM security in the hybrid framework. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 461–480. Springer, Heidelberg (Feb 2014)
9. Dent, A.W.: A designer's guide to KEMs. In: Paterson, K.G. (ed.) 9th IMA International Conference on Cryptography and Coding. LNCS, vol. 2898, pp. 133–151. Springer, Heidelberg (Dec 2003)
10. 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)
11. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th FOCS. pp. 293–302. IEEE Computer Society Press (Oct 2008)
12. 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)
13. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online authenticated-encryption and its nonce-reuse misuse-resistance. To appear in proceedings of CRYPTO 2015 (2015), http://eprint.iacr.org/2015/189
14. 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)
15. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri [3], pp. 98–107
16. 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)
17. 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)
18. Sleevi, R., Watson, M.: Web cryptography api. W3C Candidate Recommendation (2014), http://www.w3.org/TR/WebCryptoAPI/
19. Vaudenay, S.: Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS ... In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 534–546. Springer, Heidelberg (Apr / May 2002)