A preliminary version of this paper appears in *Advances in Cryptology - EUROCRYPT '12 Proceedings*. Lecture Notes in Computer Science, Vol. 7237, pp. 682–699, Pointcheval, D.; Johansson T. eds., Springer, 2012. This is the full version.

Security of Symmetric Encryption in the Presence of Ciphertext Fragmentation[*]

ALEXANDRA BOLDYREVA[†] JEAN PAUL DEGABRIELE[‡] KENNETH G. PATERSON[§]
MARTIJN STAM[¶]

February 17, 2015

Abstract

In recent years, a number of standardized symmetric encryption schemes have fallen foul of attacks exploiting the fact that in some real world scenarios ciphertexts can be delivered in a fragmented fashion. We initiate the first general and formal study of the security of symmetric encryption against such attacks. We extend the SSH-specific work of Paterson and Watson (Eurocrypt 2010) to develop security models for the fragmented setting. We also develop security models to formalize the additional desirable properties of ciphertext boundary hiding and robustness against Denial-of-Service (DoS) attacks for schemes in this setting. We illustrate the utility of each of our models via efficient constructions for schemes using only standard cryptographic components, including constructions that simultaneously achieve confidentiality, ciphertext boundary hiding and DoS robustness.

[*]This work was supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

[†]Georgia Institute of Technology, USA. E-mail: `sasha@gatech.edu`. This author supported by NSF awards CNS-0831184 and TWC-1318511.

[‡]Royal Holloway, University of London, UK. E-mail: `jean.degabriele@rhul.ac.uk`. This author supported in part by EPSRC Leadership Fellowship EP/H005455/1, and in part by Vodafone Group Services Limited, a Thomas Holloway Research Studentship, and the Strategic Educational Pathways Scholarship Scheme (Malta), part-financed by the European Union European Social Fund.

[§]Royal Holloway, University of London, UK. E-mail: `Kenny.Paterson@rhul.ac.uk`. This author supported by EPSRC Leadership Fellowship EP/H005455/1.

[¶]University of Bristol, UK. E-mail: `stam@cs.bris.ac.uk`.

Contents

1 Introduction

The provable security approach, which has become a de facto standard in modern cryptography, allows us to build cryptographic schemes with security guarantees. Many widely deployed protocols have been proven secure under reasonable assumptions and the standards bodies, such as NIST, require proofs to accompany new submissions. However, there is still a disconnect between theory and practice. Deployed schemes do get compromised sometimes, despite the existence of proofs guaranteeing security. There is no contradiction here. One of the reasons for this situation is as follows. Security proofs guarantee security *only* according to the specified security definition, which in turn may not include some adversarial capabilities possible in practice.

As one of the many examples of a kind consider SSH, one of the most widely used secure protocols. Bellare et al. [4] have formally analysed variants of SSH's Binary Packet Protocol (BPP) and showed that these variants are secure under reasonable assumptions on the protocol's building blocks. Yet a few years later, Albrecht et al. [1] presented plaintext recovery attacks against these provably secure SSH BPP variants. These attacks exploited the fact that encrypted data can be delivered to the receiver in a fragmented, byte-by-byte manner, and that the attacker can observe the receiver's behaviour at each point (in particular how long it takes to reject certain carefully crafted faulty ciphertexts). On the other hand, formal security definitions, including the one used to prove SSH secure, traditionally treat plaintexts and ciphertexts as *atomic*, meaning that the entire ciphertext is offered for decryption and a plaintext (or error symbol) is instantly returned.

To bridge this gap between theory and practice and to have schemes with security guarantees that hold not only on paper but also in reality, one has to design security definitions which are integrated better with the environments in which the protocols are deployed. Paterson and Watson [16] recently took a first step in this direction by showing that certain SSH BPP variants meet a newly introduced security notion that takes the aforementioned attacks into account. However, their security notion is heavily intertwined with the SSH BPP specification and too complex to be extended easily to apply to different schemes. We provide a more detailed critique of this precursor [16] in Appendix A.

1.1 Overview of Contributions

In this work we seek to strike the right balance between two conflicting aims: keeping the generality and simplicity of traditional security definitions for symmetric encryption; and developing a framework that can be used to provide *meaningful* provable security analyses of practical schemes when deployed in environments that permit ciphertext fragmentation attacks.

To this end, we initiate a general study of security of symmetric encryption schemes against fragmentation attacks, not only in terms of message privacy, but also in terms of length-hiding and prevention of fragmentation-enabled Denial-of-Service (DoS) attack against the receiver. To the best of our knowledge, the latter two goals for encryption, i.e. length-hiding (or, more precisely, hiding ciphertext boundaries in a ciphertext stream) and DoS prevention have not been previously studied, partly because the corresponding threats are not present if encryption is treated as being atomic. The adversarial capabilities we define are general enough to model a wide class of fragmentation attacks, including but not limited to the ones considered by Paterson and Watson [16].

We complement our new security definitions with efficient cryptographic constructions based on standard primitives meeting the new goals. While it may be relatively easy to achieve each security goal independently, it transpires that it is not straightforward to achieve two or three of the aforementioned goals simultaneously and one of our schemes is the first to do so. For comparison, Albrecht et al. [1] already observed that SSH variants fail to meet ciphertext boundary-hiding and DoS robustness under active attacks (the observation carries over to the scheme proved confidential against fragmentation attacks by Paterson and Watson [16]).

Let us now describe our focus and results in a little more detail.

Data fragmentation. Data sent over networks is often fragmented, meaning that it is broken up into smaller pieces, or packets. If the data is encrypted, the receiver first has to determine what constitutes a complete ciphertext in order to decrypt it and obtain the underlying message. (Unless the protocol requires on-the-fly decryption. This, however, is known to reduce security [14].) Reconstruction of the original ciphertext by the receiver can be accomplished by various methods. For example, SSH uses a length field that tells the receiver how many bytes are needed before the complete ciphertext has arrived; this length field is encrypted, ostensibly to increase the security of the protocol against traffic analysis. As another example, IPsec handles fragmentation of packets using fields in the IP header: the MF flag (indicating that more fragments follow), the fragment offset field (used to reorder the fragments), and the identification field (to identify which fragments belong to which packets).

During normal transmission, application-layer messages may be fragmented as required by the lower-layer protocols and the specifics of the physical networks over which they are routed. In general this cannot be known a priori and one needs to support arbitrary fragmentation patterns. Interestingly, the mechanisms in place for supporting fragmentation may be subverted by an adversary to mount specialised attacks on secure protocols. Most surprisingly, tampering with fragmentation can undermine message privacy, but it can also result in Denial of Service and aid traffic analysis. We already mentioned an attack of this kind on SSH [1]. Another example is an attack by Degabriele and Paterson [8] on the IPsec protocol. [1]

Syntax for encryption supporting fragmentation. We start our analysis with defining encryption in the presence of fragmentation. A *symmetric encryption scheme supporting fragmentation* is defined similarly to a regular atomic (fragmentation devoid) encryption scheme, except the decryption algorithm is always stateful, mainly to model decryption algorithms that may, for example, combine data coming from multiple ciphertext fragments before outputting any plaintext. In addition to state, decryption takes input fragments, one-by-one. Depending on the scheme, the minimal fragment length can be one bit, a byte or a block (of some fixed length). The correctness requirement is defined more intricately than that for atomic encryption. It requires that regardless of how one fragments the ciphertexts, upon decryption, the original messages are returned, in correct order, with correct message boundaries indicated.

When considering ciphertext fragmentation, a stateful encryption scheme becomes a more natural choice. Intuitively, the decryption algorithm will need to maintain some form of buffer in which it gathers ciphertext fragments prior to decrypting the reconstructed ciphertext. Accordingly, in this work we will by default consider encryption schemes that have a stateful decryption algorithm. If no further constraints are imposed, we will generally simply refer to such schemes as *stateful*. In the atomic setting it is well known that stateful schemes can meet stronger security notions than stateless ones. In particular stateful schemes can protect against replay and reordering attacks [4]. To translate this distinction to our setting, we define a subclass of schemes that have a 'minimal' state, serving as an analogue of stateless schemes in the atomic setting. Namely, the decryption algorithm is still stateful, but the state is restricted to behave as a buffer and hence

[1]While similar security issues arise for IPsec, its details are much more intricate and so it does not serve as such a good illustrative example as SSH. In particular, IPsec interacts heavily with IP (for the relevant cases it actually interacts with multiple instances of IP) and the two have to be considered jointly as one protocol. Our treatment focuses primarily on higher-layer protocols like SSH and TLS, but it applies just as well to the case of IPsec.

only serves to support ciphertext fragmentation. We call such schemes *stateless beyond buffering (SBB)*. See Sections 3.1 and 3.2 for the definitions.

Message privacy in the presence of fragmentation. We observe that fragmentation becomes relevant for security only in the case of chosen-ciphertext attacks (CCA). We extend the existing IND-CCA security notion and its stateful variant IND-sfCCA [4] to the case of ciphertext fragmentation which we denote as IND-sbbCFA and IND-sfCFA respectively.

Recall that for IND-sfCCA there is no restriction on the decryption queries, but if the adversary forwards the challenge ciphertexts returned by the left-or-right encryption oracle to the decryption oracle in order, this is considered in-sync, and the decryption output is suppressed. Otherwise, it is declared out-of-sync, and the decryptions are returned to the adversary. This allows the adversary to advance the state of both encryption and decryption algorithms to potentially favourable values. When dealing with fragments, the challenge is to decide when to enter the out-of-sync state. We opted to delineate the mark where the queries become out-of-sync to coincide with a ciphertext boundary rather than a fragment boundary. This conservative choice results in a stronger security notion. We provide our IND-sfCFA definition and more discussion in Section 4.1.

When considering a weaker privacy notion, one that does not protect against replay and reordering attacks, more difficulties emerge. Specifically, parsing the fragment stream for challenge ciphertexts becomes particularly daunting. To enable effective parsing, we resort to certain properties of SBB schemes, and accordingly our weaker privacy notion (IND-sbbCFA) is only meaningful for SBB schemes. A formal definition and more discussion can be found in Section 4.2.

Ciphertext boundary hiding. It is conventional wisdom in cryptographic security definitions that an encryption scheme is allowed to leak the length of the ciphertext; it is often regarded as inevitable. However, real schemes do try to hide the lengths of encrypted messages, with a view to frustrating traffic analysis based on these lengths. This is generally achieved in practice by two distinct mechanisms.

Firstly, an encryption scheme for which the ciphertext length does *not* deterministically depend on the message length may be used (e.g. by using variable-length padding). The SSH Binary Packet Protocol and the TLS Record Protocol both adopt this approach. This mechanism has recently received attention from differing perspectives [15, 18]. Secondly, an encryption scheme may be designed in such a way that it is hard to distinguish where the boundaries between ciphertexts lie in a *stream* of ciphertexts. TLS, with its explicit length field in the header of each TLS Record Protocol message, does not achieve this. But SSH's Binary Packet Protocol (BPP) does attempt to achieve boundary hiding. This necessitated the introduction of an encrypted length field in SSH, which is used by the receiver to determine how many bytes are required before a complete ciphertext has arrived and a MAC on the plaintext can be checked. However, this design decision, coupled with the use of CBC mode encryption, is precisely what enabled recent fragmentation attacks against SSH [1]. Thus having boundary hiding as a security goal can act in opposition to achieving other, more standard security goals.

In this work we focus on the second mechanism and the goal of ciphertext boundary hiding. We assume that given a message, the resulting ciphertext length is uniquely determined. In Section 5, we formalize the goal of boundary hiding for ciphertext streams. We give definitions for both the passive and the active adversary cases, which we call BH-CPA, BH-sfCFA and BH-sbbCFA (the latter refers to the SBB setting). The passive case is very common in the traffic analysis literature. Here the adversary merely monitors encrypted traffic and tries to infer information from ciphertext lengths and other information such as network packet timings, but without giving away its presence by actively modifying network traffic. By hiding the ciphertext boundaries, the adversary no longer has access to fine-grained ciphertext lengths (our solution of course does not help to hide the total volume being sent). We also define boundary hiding in the active case and find out that it is much more challenging to achieve.

Denial-of-Service. Next, we shift attention to preventing fragmentation-related Denial-of-Service (DoS) attacks against the receiver. This is, to the best of our knowledge, the first formal treatment of DoS prevention as a property of encryption. For an example of such an attack, consider the SSH-CTR scheme (see [16] for a description) and the adversary who changes the length field that occupies the first 32 bits of plaintext by bit flipping in the ciphertext. If the length is maliciously increased to a very large value (say, $2^{32} - 1$, the maximum possible value for a 32-bit field), then the receiver will continue listening for ciphertext fragments awaiting message completion, until 2^{32} bytes of data have been received. Only then will SSH-CTR's MAC verification be conducted and the message rejected. The application (or user) receiving data from the SSH connection experiences this as an SSH connection hang, a form of Denial-of-Service.

We provide security definitions for the stateful and SBB settings, DOS-sfCFA and DOS-sbbCFA resp., for DoS attacks in Section 6 that are sufficiently flexible to capture the SSH attack and others like it. Essentially, we measure the attacker's ability to create a sequence of ciphertext fragments for which the decryption algorithm of a scheme does not output any message or failure symbol within a reasonable timeframe, measured in terms of the number of bits submitted to a decryption oracle by the adversary. Accordingly we parametrise our security notions with an integer n indicating the maximum sequence of bits which do not yield any output that a reasonable adversary may produce. The DoS mitigation techniques employed by SSH and TLS require limiting the maximum supported message size to a comparable value in order to be n-DOS-sfCFA secure. We manage to surpass this limitation.

Constructions and their security. So far, our emphasis has been on developing security models and notions. However, as we proceed, we demonstrate how each of the security notions we introduce can be met in practice by efficient schemes using only standard symmetric components. These constructions are illustrative rather than definitive.

First we give a simple transformation that converts any secure atomic scheme into a secure scheme supporting fragmentation. One of the challenges that has to be overcome is ensuring correct decryption of the fragmented scheme. We solve this by applying *instantaneously decodable postprocessing* to the ciphertexts, where in the simplest of cases this can be instantiated with a prefix-free encoding scheme. This allows the decrypting algorithm to correctly parse a concatenation of ciphertexts into discrete ciphertexts which it can then decrypt in an atomic fashion. In addition, allowing the instantaneously decodable postprocessing to be keyed permits the ciphertext boundaries to be hidden from a passive adversary, i.e. it achieves BH-CPA security. Achieving BH-sfCFA requires more work. We show that this can be done at the same time as achieving DoS security we discuss next. In the SBB setting, however, we will only achieve BH-CPA security. This appears to be an inherent limitation of the SBB setting: in fact, in Section B, we show that no practically relevant SBB scheme can simultaneously meet our BH-sbbCFA and DOS-sbbCFA security notions.

Our idea for DoS prevention in the stateful setting is to break the ciphertexts into equal-sized segments and authenticate each of them individually. In our construction, the sender and receiver keep a state which contains a message and a segment number. The encryption algorithm MACs this state together with the encryption of the segment, but the state does not have to be transmitted, as the receiver maintains it for himself. Each segment is encoded using a bit flag to indicate the last segment in a message. The scheme also achieves BH-sfCFA security.

	IND-sfCFA	BH-CPA	BH-sfCFA	n-DOS-sfCFA $n < \max_{m \in \mathcal{M}}(\lvert m \rvert)$
SSH-CBC	✗	✓	✗	✗
SSH-CTR	✓	✓	✗	✗
IDP (Sec. 7.1)	✓	✓	✗	✗
$\mathcal{IM}$ (Sec. 7.2)	✓	✓	✓	✓
	IND-sbbCFA	BH-CPA	BH-sbbCFA	n-DOS-sbbCFA $n < \max_{m \in \mathcal{M}}(\lvert m \rvert)$
$\mathcal{IM}^*$ (Sec. 7.3)	✓	✓	✗	✓

Table 1: Security comparison of encryption schemes supporting fragmentation.

In the SBB setting we also use the idea of authenticating the ciphertext segments. However, the solution becomes more complex, as we cannot keep message and segment numbers as state and it is not efficient to keep them as part of the segments. To prevent re-ordering attacks we need to authenticate the previous segments as well. To improve efficiency we chain the tags instead, i.e. we compute tags over the concatenation of the segment and the previous tag value. In contrast to the stateful case, the scheme is only BH-CPA secure.

Table 1 summarizes the security properties achieved by our main constructions for the stateful and SBB settings. For comparison we also cast prior work on the SSH-CBC and SSH-CTR schemes [4, 16] into our framework. For construction definitions and further discussions, see the referenced sections. We note that the scheme InterMAC is able to simultaneously achieve all three of our active security notions IND-sfCFA, BH-sfCFA, and n-DOS-sfCFA without limiting the maximum message size.

1.2 Further Related Work

Our fragmented approach bears more than a passing resemblance to work on on-line encryption [2, 3, 6, 10, 11, 12, 14]. However, whereas the on-line setting concerns a single continuous message and ciphertext, with each block of plaintext leading to a block of ciphertext being output during encryption (and vice-versa during decryption), our setting concerns atomic encryption (reflecting how many secure protocols operate) but allows fragmented decryption of ciphertexts. Moreover, we extensively treat the case of active adversaries, a topic that has not achieved much attention in the on-line literature, and we consider more than just confidentiality security notions. This said, our ultimate construction, InterMAC, can be seen as a kind of online scheme with a large block size.

1.3 Differences From the Proceedings Version

This is a revised and augmented version of a paper with the same title that appeared in the EUROCRYPT 2012 proceedings. We warn the reader that some of the naming and notation has been altered. In particular the prefix-free postprocessing construction has been renamed to instantaneously decodable postprocessing. The Indistinguishability and Boundary-Hiding security definitions are essentially unchanged, only the presentation is different. The same holds for the InterMAC construction. On the other hand, the Denial-of-Service security definition has been strengthened. Specifically, we previously required the adversary to produce the winning sequence of ciphertext fragments (an n-bit long sequence of ciphertext fragments that produce no output upon decryption, see Section 6) as soon as his decryption queries became out-of-sync. We no longer make this restriction: we allow the winning fragment sequence to appear at any point after the decryption queries become out-of-sync.

2 Notation

Unless otherwise stated, an algorithm may be randomised. An adversary is an algorithm. For any algorithm $\mathcal{A}$ we use $y \leftarrow \mathcal{A}(x_1, x_2, \dots)$ to denote executing $\mathcal{A}$ with fresh coins on inputs $x_1, x_2, \dots$ and assigning its output to y. If $\mathcal{S}$ is a set then $|\mathcal{S}|$ denotes its size, and $y \leftarrow_\$ \mathcal{S}$ denotes the process of selecting an element from $\mathcal{S}$ uniformly at random and assigning it to y.

The set of all finite binary strings is denoted by $\{0,1\}^*$. For any positive integer n and bit b, we denote by b^n the string of n consecutive b's and $\{0,1\}^n$ represents the set of all binary strings of length n. The empty string is represented by ε. For any two strings w and z and positive integers i and j, $w \parallel z$ denotes their concatenation, $w \oplus z$ denotes their bitwise XOR, $w \diamond z$ denotes the greatest common prefix of w and z, $w \,\%\, z$ denotes the remainder string of w with respect to $w \diamond z$ (i.e. $w = w \diamond z \parallel w \,\%\, z$), and $|w|$ denotes the length of w. Accordingly, we can express that w is a prefix of z as $w \diamond z = w$. A set $S \subset \{0,1\}^*$ is called prefix-free if for all distinct $w, z \in S$ it holds that w is not a prefix of z. Unless stated otherwise, $w[i]$ denotes the i^{th} bit of w, and $w[i,j]$ denotes the substring $w[i] \parallel w[i+1] \parallel \dots \parallel w[j]$. For any $n \in \mathbb{N}$, and any vector of strings $\mathbf{w} = [w_1, w_2, \dots, w_n]$, we define the concatenation operator as $||(\mathbf{w}) = w_1 \parallel w_2 \parallel \dots \parallel w_n$. If j is a non-negative integer, then $\langle j \rangle_\ell$ denotes the unsigned ℓ-bit binary representation of j. Accordingly $\langle \cdot \rangle^{-1}$ represents the inverse mapping which maps strings of any length to $\mathbb{N}$. If w is an ℓ-bit string and i is an integer we use $w + i$ as shorthand for $\langle \langle w \rangle^{-1} + i \bmod 2^\ell \rangle_\ell$. Finally, a list $\mathtt{L}$ is a totally ordered set. We use $()$ to denote the empty list, and use $\mathtt{L}_i$ to denote the i^{th} element of the list.

3 Symmetric Encryption Supporting Fragmentation

3.1 Unified Syntax

Morphology. We extend the standard definition of symmetric encryption for the case of fragmented ciphertexts. For fragmentation to make sense, we will restrict our attention to ciphertexts that are strings, so $\mathcal{C} = \{0,1\}^*$. We assume that the message space consists of strings, so $\mathcal{M} = \{0,1\}^*$. In addition we will allow schemes to have multiple errors (cf. [5]). The move to fragmentation results in some complications. For instance, a single ciphertext can be split up in multiple fragments (Fig. 1, requiring recombination of the decryptions of the various fragments) or a single fragment can contain multiple ciphertexts (Fig. 2, where one might like to decrypt to a list of messages).

Definition 3.1. [Symmetric encryption scheme supporting fragmentation]

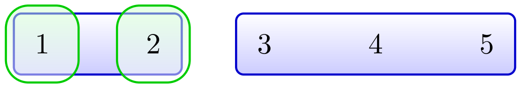

Figure 1: A single ciphertext $c_1 = (12)$ cut up in multiple fragments $f_1 = (1)$ and $f_2 = (2)$.

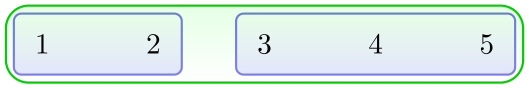

Figure 2: A single fragment $f_1 = (12345)$ spanning multiple ciphertexts $c_1 = (12)$ and $c_2 = (345)$.

A *symmetric encryption scheme supporting fragmentation* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with associated *message space* $\mathcal{M} \subseteq \{0,1\}^*$, *ciphertext space* $\mathcal{C} \subseteq \{0,1\}^*$ and *error messages* $\mathcal{S}_\perp$ is defined by three algorithms:

- The randomised *key generation* algorithm $\mathcal{K}$ returns a secret key K and initial states σ_0 and ϱ_0.
- The randomised and stateful *encryption* algorithm

  $$\mathcal{E} : \mathcal{K} \times \mathcal{M} \times \Sigma \to \mathcal{C} \times \Sigma$$

  takes as input the secret key $K \in \mathcal{K}$, a plaintext $m \in \mathcal{M}$, and the current encryption state $\sigma \in \Sigma$, and returns a ciphertext in $\mathcal{C}$ together with an updated state. For any $\ell \in \mathbb{N}$ and any $\mathbf{m} = [m_1, \ldots, m_\ell] \in \mathcal{M}^\ell$, we write $(\mathbf{c}, \sigma) \leftarrow \mathcal{E}_K(\mathbf{m}, \sigma_0)$ as shorthand for $(c_1, \sigma_1) \leftarrow \mathcal{E}_K(m_1, \sigma_0)$, $(c_2, \sigma_2) \leftarrow \mathcal{E}_K(m_2, \sigma_1), \ldots (c_\ell, \sigma_\ell) \leftarrow \mathcal{E}_K(m_\ell, \sigma_{\ell-1})$ where $\mathbf{c} = [c_1, \ldots, c_\ell]$ and $\sigma = \sigma_\ell$.
- The deterministic and stateful *decryption* algorithm

  $$\mathcal{D} : \mathcal{K} \times \{0,1\}^* \times \Sigma \to (\{0,1\} \cup \{\P\} \cup \mathcal{S}_\perp)^* \times \Sigma$$

  takes the secret key K, a ciphertext fragment $f \in \{0,1\}^*$, and the current decryption state ϱ to return the corresponding plaintext fragment $m \in (\{0,1\} \cup \{\P\} \cup \mathcal{S}_\perp)^*$ together with the updated state ϱ. For any $\ell \in \mathbb{N}$ and any $\mathbf{f} = [f_1, \ldots, f_\ell] \in (\{0,1\}^*)^\ell$, we write $(m, \varrho) \leftarrow \mathcal{D}_K(\mathbf{f}, \varrho_0)$ as shorthand for $(m_1, \varrho_1) \leftarrow \mathcal{D}_K(f_1, \varrho_0)$, $(m_2, \varrho_2) \leftarrow \mathcal{D}_K(f_2, \varrho_1), \ldots (m_\ell, \varrho_\ell) \leftarrow \mathcal{D}_K(f_\ell, \varrho_{\ell-1})$, where $m = m_1 \parallel \ldots \parallel m_\ell$ and $\varrho = \varrho_\ell$.

This definition requires a little unpacking. Firstly, and in contrast to the usual definitions, our syntax defines both encryption and decryption to be stateful algorithms. This does not result in any loss of generality: both encryption and decryption can be made stateless by having $\mathcal{K}$ to always return the empty string for the corresponding initial state, and let the algorithm ignore (i.e. never update) the state. We adopt this syntax because supporting ciphertext fragmentation inherently requires the decryption algorithm to be stateful. We will elaborate more on this later on in this section and in Section 3.2. Furthermore, our primary motivation for studying ciphertext fragmentation are secure protocols such as TLS, SSH, IPsec, and DTLS. These protocols include mechanisms (other than for the purpose of supporting ciphertext fragmentation) that render their decryption stateful, and generally support ciphersuites with stateful encryption algorithms.

Secondly, note that the decryption algorithm is assumed to be able to handle ciphertexts which decrypt to multiple plaintext messages, or to a mixture of plaintexts and error symbols, or possibly to nothing at all (perhaps because the input ciphertext is insufficient to enable decryption to yet output anything, giving

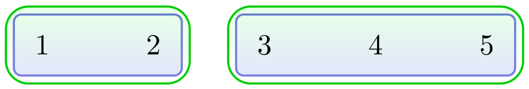

Figure 3: Fragments $f_1 = (12)$ and $f_2 = (345)$ coincide exactly with the two ciphertexts $c_1 = (12)$ and $c_2 = (345)$.

a significant difference from the atomic setting where decryption always outputs something). We use $¶ \notin \{0,1\} \cup \mathcal{S}_\perp$ to denote the end of plaintext messages, enabling an application making use of the decryption algorithm to parse the output uniquely into a sequence of elements of $\{0,1\}^*$ and errors from $\mathcal{S}_\perp$. Our introduction of an explicit symbol ¶ to help delineate messages during decryption seems novel. This is not because our solution is in any way innovative, but rather because the problem does not arise in earlier works.

Thirdly, note that, when failing, the decryption algorithm can output one of possibly many error messages from the set $\mathcal{S}_\perp$. This reflects the fact that real schemes may fail in more than one way, with the different failure modes being visible to both legitimate users and adversaries. Boldyreva et al. [5] provide a more in-depth examination of dealing with multiple error messages in the context of symmetric encryption.

While we enforce that from a decryption of a sequence of ciphertext fragments, the corresponding message boundaries are easy to distinguish, we make no such requirement for ciphertexts. Indeed, given a sequence of ciphertext fragments, it will not be a priori clear what the constituent ciphertexts are (and in fact, in Section 5, we want to model schemes which hide these boundaries as a security goal). Looking ahead, the absence of clear ciphertext boundaries (in a sequence of fragments) will cause challenging parsing problems for our CCA definitions: in order to 'forbid' decryption of the challenge ciphertext, a prerequisite is that this challenge ciphertext can be located accurately in the sequence of ciphertext fragments!

For simplicity's sake we did not to incorporate the nonce-based approach from [17] in our syntax, yet it should be possible to extend most of our work to that setting.

Correctness. If a single message is encrypted and the corresponding ciphertext is subsequently decrypted, we expect that the message is returned. When multiple messages are encrypted and the fragments correspond *exactly* to the ciphertext (Fig. 3), again we expect to retrieve the original messages.

However, we expect something stronger, namely that regardless of how we fragment the ciphertext(s), the original message(s) are returned. For instance in the situation depicted in Fig. 1 two ciphertexts $c_1 = (12)$ and $c_2 = (345)$ are produced by the encryption oracle, and the adversary subsequently submits fragments $f_1 = (1)$ and $f_2 = (2)$ to its decryption oracle. We require that after reception of the second fragment (or earlier), the ciphertext c_1 gets decrypted (formalised by the correct message being output). Similarly, in Fig. 2, after reception of the single fragment spanning two ciphertexts, we expect both messages to be returned (with correct message boundaries indicated).

Finally, we require correct decryption, even when an extra string is added to the original (string of) ciphertexts. This forces correct decryption once a complete valid ciphertext has been received, even if it is followed by an invalid ciphertext fragment. For instance, in the situation depicted in Fig. 5 two ciphertexts $c_1 = (12)$ and $c_2 = (345)$ are produced by the encryption oracle, the adversary subsequently submits fragments $f_1 = (1)$ and $f_2 = (234'5')$ to its decryption oracle, and we still want to see the first ciphertext decrypted properly.

With this intuition in mind, we are almost ready to give our definition of correctness for a symmetric encryption scheme supporting fragmentation. We first define a map $¶ : (\{0,1\}^* \cup \mathcal{S}_\perp)^* \to (\{0,1\} \cup \{¶\} \cup \mathcal{S}_\perp)^*$ by $¶(m_1, \dots, m_\ell) = m_1 \parallel ¶ \parallel \dots ¶ \parallel m_\ell \parallel ¶$. Note that ¶ is injective but not surjective.

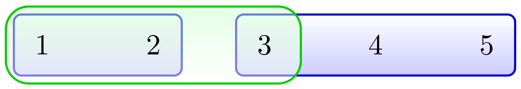

Figure 4: Correct decryption for overly long fragments: Given valid ciphertexts $c_1 = (12)$ and $c_2 = (345)$ and fragment $f_1 = (123)$, what should the decryption of f_1 be?

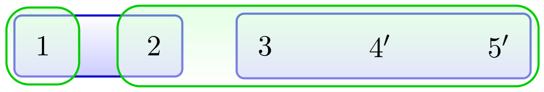

Figure 5: Two consecutive fragments $f_1 = (1)$ and $f_2 = (234'5')$. The second fragment completes the first ciphertext $c_1 = (12)$, so we expect that to be decrypted at this point, even though ciphertext $c_2 = (345)$ in the second fragment has been modified to produce a possibly invalid ciphertext.

Definition 3.2. [Correctness Requirement] For all (K, σ_0, ϱ_0) that can be output by $\mathcal{K}$ and for all $\mathbf{m} \in \mathcal{M}^*$ and $\mathbf{f} \in (\{0,1\}^*)^*$, it holds (with probability 1) that if $(\mathbf{c}, \sigma) \leftarrow \mathcal{E}_K(\mathbf{m}, \sigma_0)$ and $||(\mathbf{c})$ prefixes $||(\mathbf{f})$, and if $(m', \varrho) \leftarrow \mathcal{D}_K(\mathbf{f}, \varrho_0)$ then m' is prefixed by $\P(\mathbf{m})$.

Alternatives. Our choice for correctness (Definition 3.2) might seem natural, but it is not the only way to define it. Certainly, if a single, honestly generated ciphertext is cut up into multiple fragments, then decrypting all those fragments ought to result in the original message. This extends to a situation where (the concatenation of) multiple ciphertexts is split up into fragments in such a way that *every* ciphertext boundary coincides with a fragment boundary (this implies that every fragment is a substring of a single ciphertext). However, when allowing a single fragment to extend over multiple ciphertexts (see Fig. 4 for an example), it is not immediately clear what 'correct' entails. Let us briefly consider three possible interpretations.

Fault: The ciphertext is deemed invalid, and $\perp$ is returned.

Flush: The message is returned, and any surplus ciphertext is ignored.

Buffer: The message is returned, and any surplus ciphertext is considered as starting a new ciphertext (buffering).

We have opted for a strict version of the final interpretation in our correctness definition, which intuitively requires some sort of buffering to take place in the decryption algorithm. Thus our choice of definition for correctness inherently requires any scheme that supports fragmentation to have a stateful decryption algorithm. SSH is a prime example of a stateful scheme that buffers (although it keeps more state than just the buffer). Next we classify two different degrees of statefulness that a scheme may have.

3.2 Degrees of Statefulness

Protocols like TLS, SSH, IPsec, and DTLS, use authenticated sequence numbers during decryption to detect replays and reorderings of ciphertexts. Additionally, in SSH and TLS when certain errors are detected during decryption the secure session is torn down and the session keys are destroyed. This can be modelled by maintaining a flag as part of the decryption state, setting the flag once a first error is encountered, and always outputting a failure symbol once the flag is set. When appropriately incorporated in (atomic) symmetric encryption schemes, these stateful techniques permit the scheme to meet stronger security notions such as the INT-sfCTXT and IND-sfCCA notions of [4]. When considering ciphertext fragmentation, in light of our correctness requirement, some form of buffering is generally required on the decryption side. Albeit intended for a different purpose, such a buffer is technically a state that is maintained by the decryption algorithm across calls. Thus all schemes that we will consider will be stateful, and as a special case we will consider schemes that employ only a 'minimal' form of state necessary to support fragmentation. For each notion of security we will first present a 'strong' variant following similar ideas to the security notions in [4]. These notions will generally be met only by schemes which maintain a non-minimal state. Then we will present weakened variants of these notions which can be met by schemes within the special subclass having minimal state.

Stateful schemes. This includes all schemes supporting ciphertext fragmentation, and no restriction is imposed on the nature of the state that is maintained by the decryption algorithm. The encryption algorithm may or may not be stateful. This covers most practical encryption schemes, which, in the non-fragmented scenario, would normally be considered stateful. For instance, this can be used to model the situation where encryption and decryption are both based on a counter that increases depending on the number of messages or ciphertexts processed.

Stateless Beyond Buffering (SBB) schemes. This is a subclass of the above category, which is intuitively an extension of standard (atomic), stateless encryption schemes that makes handling fragmented ciphertexts possible. Namely, we specify three properties which intuitively capture the behaviour of a buffer, and require that the decryption state satisfy these properties. Our formulation allows us to identify states that essentially act as buffers, without imposing any restrictions on the state's internals or format. Again the encryption algorithm may or may not be stateful. This is in line with [4] where the statefulness of a scheme is determined solely by the statefulness of the decryption algorithm. More formally, we have:

Definition 3.3. [Stateless Beyond Buffering (SBB)] A symmetric encryption scheme supporting fragmentation is called *stateless beyond buffering* if it is correct (Definition 3.2) and satisfies the following additional conditions

1. The initial decryption state is empty, that is for all (K, σ_0, ϱ_0) that can be output by $\mathcal{K}$, $\varrho_0 = \varepsilon$; for simplicity's sake, we will often simply write $(K, \sigma) \leftarrow \mathcal{K}$ for SBB schemes.
2. The decryption state is empty after decryption of each ciphertext obtained from encryption, i.e. for all K that can be output by $\mathcal{K}$, for all $\sigma \in \Sigma$, for all $m \in \mathcal{M}$, it holds (with probability 1) that if $(c, \sigma) \leftarrow \mathcal{E}_K(m, \sigma)$ and if $(m', \varrho) \leftarrow \mathcal{D}_K(c, \varepsilon)$, then $\varrho = \varepsilon$.
3. The scheme satisfies *literal decryption:* for all $K \in \mathcal{K}$ and for all $\mathbf{f} = (f_1, \ldots, f_\ell)$, when $f' = f_1 \parallel \ldots \parallel f_\ell$, then $\mathcal{D}_K(\mathbf{f}, \varepsilon) = \mathcal{D}_K(f', \varepsilon)$.

The first condition is straightforward and its purpose is to ensure that the decryption algorithm is not initialised with any state information. The second condition says that for legitimately generated ciphertexts, the decryption state is flushed when the end of a ciphertext is detected. Put differently, this condition ensures that no state information is maintained across ciphertexts, i.e. the decryption of one ciphertext does not depend on previous ciphertexts. However this 'stateless' behaviour is only guaranteed as long as the ciphertexts are generated by the encryption algorithm. In particular for an adversarially-generated sequence of ciphertext fragments, depending on the scheme at hand, a number of outcomes are possible. After a few ciphertext fragments the decryption algorithm may detect the end of a ciphertext and return ¶, possibly following $\perp \in \mathcal{S}_\perp$ if the ciphertext was deemed invalid, or after some plaintext if the ciphertext was understood to be valid. Alternatively the decryption may never recover, in the sense that it will never return ¶ but possibly it may return a sequence of outputs in $(\{0,1\} \cup \mathcal{S}_\perp)^*$. From the second property it follows that state can only be maintained across fragments belonging to the same ciphertext. The literal decryption property then says that the decryption state will not (or does not need to) keep track of how the ciphertext was fragmented, since it will not affect the output of the decryption algorithm.

4 Message Privacy of Schemes Supporting Fragmentation

4.1 The Stateful Notion

When considering the security of a scheme supporting fragmentation, the first thing to note is that fragmentation matters only in the CCA setting: if there is no decryption oracle, then whether decryption is fragmented or atomic is immaterial to the security of the scheme. In the context of fragmentation, we will replace the usual notion of chosen-ciphertext attacks by chosen-fragment attacks (CFA). Our first notion, IND-sfCFA is tailored for stateful schemes and it is inspired by Bellare et al.'s notion of IND-sfCCA (for atomic schemes) [4]. Recall that for IND-sfCCA, an adversary has unlimited access to the decryption oracle; there are no 'prohibited' queries. Instead, to avoid trivial attacks (by the adversary simply relaying its challenge ciphertext for decryption) a syncing mechanism is used. Initially the decryption oracle is in-sync and its output (to the adversary) will be suppressed. Only when the adversary causes the decryption oracle to be out-of-sync (by deviating from the ciphertext stream output by the encryption oracle) will the purported plaintexts (or error messages) be returned.

For atomic schemes, this is relatively straightforward to define, but for schemes supporting fragmentation, some ambiguity arises. Consider again the scenario sketched in Fig. 5. The first fragment is in-sync and any plaintext output corresponding to it will be suppressed. In the second fragment a deviation from the challenge ciphertext stream occurs. However, *part* of the fragment is still in-sync and certainly outputting the full decryption would—mindful of the correctness requirement—reveal (part of) the plaintext (12). We will need to formalise this by officially declaring part of the fragment in-sync, and part of it out-of-sync. The ambiguity arises with regards to the boundary we should use: is sync lost already at '3' (being the first symbol of a ciphertext that is not completed properly) or only at '4' (being the first symbol of the fragment that actually deviates)?

In our definition of IND-sfCFA (Definition 4.1) we opted for the strongest interpretation, namely where synchronisation is lost at the ciphertext boundary. Since this results in synchronization potentially being lost *earlier*, the decryption oracle consequently suppresses *less* of its output, making it the stronger option.

Definition 4.1. [IND-sfCFA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a bit b, define experiment $\mathbf{Exp}_{\mathcal{SE}}^{\text{ind-sfcfa-b}}(\mathcal{A})$ as depicted in Fig. 6. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is given access to a left-or-right encryption oracle $\mathsf{LoR}(\cdot)$ and a stateful decryption oracle $\mathsf{sfDec}(\cdot)$. The stateful decryption oracle can be queried on any sequence of ciphertext fragments, but as long as the decryption queries are in sync the output will be artificially suppressed.

The adversary's goal is to output a bit b' as its guess of the challenge bit b, and the experiment returns b'

```
$\underline{\mathbf{Exp}^{\text{ind-sfcfa-b}}_{\mathcal{SE}}(\mathcal{A})}$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, j \leftarrow 0, \mathsf{sync} \leftarrow 1$
    $C \leftarrow \varepsilon, F \leftarrow \varepsilon, M \leftarrow \varepsilon$
    $\mathtt{C} \leftarrow (), \mathtt{M} \leftarrow ()$
    $b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot), \mathsf{sfDec}(\cdot)}$
    return $b'$

$\underline{\mathsf{LoR}((m_0, m_1))}$
    if $|m_0| \neq |m_1|$ then return $\lightning$
    $(c, \sigma) \leftarrow \mathcal{E}_K(m_b, \sigma)$
    $i \leftarrow i + 1, \mathtt{C}_i \leftarrow c, \mathtt{M}_i \leftarrow m_b$
    return $c$

$\underline{\mathsf{sfDec}(f)}$
    $(m, \varrho) \leftarrow \mathcal{D}_K(f, \varrho)$
    $F \leftarrow F \parallel f, M \leftarrow M \parallel m$
    if $\mathsf{sync} = 1$ then
        while $C \diamond F = C$ and $j < i$
            $j \leftarrow j + 1$
            $C \leftarrow C \parallel \mathtt{C}_j$
        if $F \diamond C = F$ then $m \leftarrow \varepsilon$
        else
            $\mathsf{sync} \leftarrow 0$
            $m' \leftarrow \P(\mathtt{M}_1, \ldots, \mathtt{M}_{j-1})$
            $m \leftarrow M \,\%\, m'$
    return $m$
```

Figure 6: Experiment to define IND-sfCFA security.

as well. The corresponding advantage of an adversary $\mathcal{A}$ is given by:

$$\mathbf{Adv}^{\text{ind-sfcfa}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{ind-sfcfa-1}}_{\mathcal{SE}}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-sfcfa-0}}_{\mathcal{SE}}(\mathcal{A}) = 1\right].$$

The scheme $\mathcal{SE}$ is said to be IND-sfCFA secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{ind-sfcfa}}_{\mathcal{SE}}(\mathcal{A})$ is small.

A few words of explanation about the workings of sfDec$(\cdot)$ are in order. Recall that $C \diamond F$ denotes the greatest common prefix of C and F. Thus the test condition $C \diamond F = C$ checks whether C is a prefix of F. The while loop starts by gathering the sequence of complete ciphertexts that have been relayed from the left-or-right oracle to the decryption oracle, concatenates them into one string, appends the subsequent ciphertext output by LoR$(\cdot)$ (if this exists), and stores the output in C. Then if F (the concatenation of all ciphertext fragments submitted for decryption) is a prefix of C, the queries are deemed to be in sync and the output is suppressed. Otherwise the sync flag is set to 0 and the output string corresponding to the first out-of-sync ciphertext and onwards is returned.

4.2 A Notion for SBB Schemes

Similarly to the stateful notions from [4], Definition 4.1 protects against attacks which replay and reorder ciphertexts. In order for a scheme to protect against such attacks it needs to maintain a decryption state across ciphertexts. Hence IND-sfCFA is 'too strong' for SBB schemes, as the second requirement of Definition 3.3 explicitly rules out the ability to maintain states across ciphertexts. Accordingly for SBB schemes we propose an analogous but weaker notion of confidentiality which does not capture replay and reordering of ciphertexts. In this setting detecting prohibited queries, which would lead to trivial win conditions, becomes challenging. In fact we resort to specific properties of SBB schemes in this definition, specifically literal decryption and the property that no state is maintained across ciphertexts. Inside the decryption oracle

```
$\underline{\mathbf{Exp}^{\text{ind-sbbcfa-b}}_{\mathcal{SE}}(\mathcal{A})}$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, F \leftarrow \varepsilon, \mathsf{C} \leftarrow ()$
    $b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot),\mathsf{Dec}(\cdot)}$
    return $b'$

$\underline{\mathsf{LoR}((m_0, m_1))}$
    if $|m_0| \neq |m_1|$ then return $\lightning$
    $(c, \sigma) \leftarrow \mathcal{E}_K(m_b, \sigma)$
    $i \leftarrow i + 1, \mathsf{C}_i \leftarrow c$
    return $c$

$\underline{\mathsf{Dec}(f)}$
    $M \leftarrow \varepsilon, \mathsf{prefix} \leftarrow 0$
    $len \leftarrow |f|$
    for $k = 1$ to $len$
        $(m, \varrho) \leftarrow \mathcal{D}_K(f[k], \varrho)$
        $m' \leftarrow m' \parallel m$
        $F \leftarrow F \parallel f[k]$
        if $\varrho = \varepsilon$ and $m'[|m'|] = \P$ then
            if $F \in \mathsf{C}$ then $m' \leftarrow \varepsilon$
            $M \leftarrow M \parallel m'$
            $F \leftarrow \varepsilon, m' \leftarrow \varepsilon$
    for all $c \in \mathsf{C}$
        if $F \diamond c = F$ then $\mathsf{prefix} \leftarrow 1$
    if $\mathsf{prefix} = 0$ then
        $M \leftarrow M \parallel m', m' \leftarrow \varepsilon$
    return $M$
```

Figure 7: Experiment to define IND-sbbCFA security.

we exploit the literal decryption property to decrypt ciphertext fragments incrementally, i.e. bit by bit. Then we can detect ciphertext boundaries by looking for a condition where the last output symbol is ¶ and the decryption state is empty. Since Definition 4.2 makes use of properties specific to SBB schemes, it only guarantees a meaningful notion of security for this subclass of schemes, and not schemes supporting fragmentation in general.

Definition 4.2. [IND-sbbCFA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation that is stateless-beyond-buffering. For an adversary $\mathcal{A}$ and a bit b, define experiment $\mathbf{Exp}^{\text{ind-sbbcfa-b}}_{\mathcal{SE}}(\mathcal{A})$ as depicted in Fig. 7. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to a left-or-right encryption oracle $\mathsf{LoR}(\cdot)$ and a decryption oracle $\mathsf{Dec}(\cdot)$. The decryption oracle can be queried on any sequence of ciphertext fragments, except that the output corresponding to a ciphertext previously output by the left-or-right oracle will be artificially suppressed.

The adversary's goal is to output a bit b' as its guess of the challenge bit b, and the experiment returns b' as well. The corresponding advantage of an adversary $\mathcal{A}$ is given by:

$$\mathbf{Adv}^{\text{ind-sbbcfa}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{ind-sbbcfa-1}}_{\mathcal{SE}}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-sbbcfa-0}}_{\mathcal{SE}}(\mathcal{A}) = 1\right].$$

The scheme $\mathcal{SE}$ is said to be IND-sbbCFA secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{ind-sbbcfa}}_{\mathcal{SE}}(\mathcal{A})$ is small.

Note that M now represents the string that is returned by the decryption oracle in response to the queried fragment f; accordingly this is always reset at the beginning. The variable F accumulates bits corresponding to a single ciphertext, and is kept to monitor whether this ciphertext was previously output by the encryption oracle. The contents of F are maintained across calls to the decryption oracle, and are only reset when a ciphertext boundary is encountered. Similarly, m' accumulates the plaintext bits corresponding to a single message. If after processing f, F does not yet contain a complete ciphertext, but it contains a prefix of a ciphertext that was previously output by the encryption oracle, the corresponding plaintext is not output but is stored in m' instead.

5 Boundary Hiding

It is conventional wisdom that an encryption scheme cannot hide entirely the message length from an adversary. In practice however, the message length can convey information about the nature of the message. For instance the IPsec attacks from [7, 8] identify ICMP error messages from their length and use this to recover the full plaintext of encrypted messages. As another example, traffic analysis has been used to derive approximate transcripts of encrypted Voice over IP (VoIP) conversations [19]. Traffic analysis is a real concern, and in practice heuristic countermeasures are commonly employed to mitigate such attacks. Practical protocols like TLS, SSH, and IPsec use variable-length padding, while IPsec additionally provides the ability to insert dummy messages/packets. A recent study by Dyer et al. [9] shows that none of the aforementioned countermeasures, together with others that have been proposed in the literature, are effective in preventing HTTP fingerprinting. However their attacks do not rely solely on ciphertext lengths.

The ability to fragment ciphertexts without affecting correct decryption may be exploited as an alternative (heuristic) means to frustrate traffic analysis. Ciphertext lengths may no longer be evident from a stream of randomly-fragmented ciphertexts flowing across a channel. However this requires the encryption scheme to not reveal ciphertext boundaries. We therefore formalise the goal of hiding ciphertext boundaries within a concatenation of ciphertexts as an intermediate security goal towards this heuristic strategy and preventing traffic analysis in general.

We give definitions for both the passive and the active adversary cases. The passive case is the one that is commonly assumed in the traffic analysis literature [9, 19]. Here the adversary merely monitors encrypted traffic and tries to infer information from ciphertext lengths and other information such as network packet timings, but without risking of giving away its presence by actively modifying network traffic. If ciphertext boundaries are not discernible to the adversary, then it can no longer determine individual ciphertext lengths, except of course, for the total volume being sent. As we will see, achieving security in the passive case is relatively straightforward. Much more challenging is achieving security in the active case. For example, it was already pointed out by Albrecht et al. [1] that SSH, while attempting to hide ciphertext boundaries, fails to do so against active, fragmented attacks (there is a simple bit-flipping attack which works irrespective of whether CBC or CTR mode encryption is used in the SSH construction).

5.1 Security Definitions

Definition 5.1. [BH-CPA and BH-sfCFA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a bit b, define experiments $\mathbf{Exp}_{\mathcal{SE}}^{\text{bh-cpa-b}}(\mathcal{A})$ and $\mathbf{Exp}_{\mathcal{SE}}^{\text{bh-sfcfa-b}}(\mathcal{A})$ as shown in Figure 8. Both experiments start by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is given access to a special left-or-right encryption oracle $\mathsf{LoR}(\cdot)$: on input two vectors of messages, either the left or the right result is returned, but with the caveat that the concatenation of ciphertexts is returned *only* if it has the same length in both worlds (but note that we do not insist that the two vectors of messages contain the same number of components). In the latter experiment the adversary is additionally given a stateful decryption oracle $\mathsf{sfDec}(\cdot)$ identical to that used in the $\mathsf{IND\text{-}sfCFA}$ experiment. The adversary can query this oracle on any sequence of ciphertext fragments, but as long as the decryption queries are

```
$\underline{\mathbf{Exp}^{\text{bh-cpa-b}}_{\mathcal{SE}}(\mathcal{A})}$ $\boxed{\underline{\mathbf{Exp}^{\text{bh-sfcfa-b}}_{\mathcal{SE}}(\mathcal{A})}}$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, j \leftarrow 0, \mathsf{sync} \leftarrow 1$
    $C \leftarrow \varepsilon, F \leftarrow \varepsilon, M \leftarrow \varepsilon$
    $\mathtt{C} \leftarrow (), \mathtt{M} \leftarrow ()$
    $b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot)}$ $\boxed{b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot),\mathsf{sfDec}(\cdot)}}$
    return $b'$

$\underline{\mathsf{LoR}((\mathbf{m}_0, \mathbf{m}_1))}$
    $\sigma_0 \leftarrow \sigma, \sigma_1 \leftarrow \sigma$
    $(\mathbf{c}_0, \sigma_0) \leftarrow \mathcal{E}_K(\mathbf{m}_0, \sigma_0)$
    $(\mathbf{c}_1, \sigma_1) \leftarrow \mathcal{E}_K(\mathbf{m}_1, \sigma_1)$
    $c_0 \leftarrow ||(\mathbf{c}_0), c_1 \leftarrow ||(\mathbf{c}_1)$
    if $|c_0| \neq |c_1|$ then return $\lightning$
    $\sigma \leftarrow \sigma_b$
    for $k = 1$ to $k = |\mathbf{c}_b|$
        $i \leftarrow i + 1$
        $\mathtt{C}_i \leftarrow \mathbf{c}_b(k), \mathtt{M}_i \leftarrow \mathbf{m}_b(k)$
    return $c_b$

$\underline{\mathsf{sfDec}(f)}$
    $(m, \varrho) \leftarrow \mathcal{D}_K(f, \varrho)$
    $F \leftarrow F \parallel f, M \leftarrow M \parallel m$
    if $\mathsf{sync} = 1$ then
        while $C \diamond F = C$ and $j < i$
            $j \leftarrow j + 1$
            $C \leftarrow C \parallel \mathtt{C}_j$
        if $F \diamond C = F$ then $m \leftarrow \varepsilon$
        else
            $\mathsf{sync} \leftarrow 0$
            $m' \leftarrow \P(\mathtt{M}_1, \ldots, \mathtt{M}_{j-1})$
            $m \leftarrow M \,\%\, m'$
    return $m$
```

Figure 8: Experiments to define BH-CPA and BH-sfCFA security. For BH-CPA the boxed code is excluded, whereas for BH-sfCFA the boxed code replaces the code adjacent to it.

in sync the output is artificially suppressed.

In both experiments, the adversary's goal is to output a bit b' as its guess of the challenge bit b, and the experiment returns b' as well. The corresponding advantages of an adversary $\mathcal{A}$ are given by:

$$\mathbf{Adv}^{\text{bh-cpa}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{bh-cpa-1}}_{\mathcal{SE}}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{bh-cpa-0}}_{\mathcal{SE}}(\mathcal{A}) = 1\right],$$

$$\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{bh-sfcfa-1}}_{\mathcal{SE}}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{bh-sfcfa-0}}_{\mathcal{SE}}(\mathcal{A}) = 1\right].$$

The scheme $\mathcal{SE}$ is said to be BH-CPA (or BH-sfCFA) secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{bh-cpa}}_{\mathcal{SE}}(\mathcal{A})$ (respectively $\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{SE}}(\mathcal{A})$) is small.

Analogously to the case of confidentiality, we can define a natural SBB variant of this notion by replacing the stateful decryption oracle sfDec$(\cdot)$ in Fig. 8 with the decryption oracle Dec$(\cdot)$ from Fig. 7. As before the resulting experiment, displayed in Fig. 9, assumes properties that are specific to SBB schemes, and hence this security notion is only meaningful for SBB schemes.

Definition 5.2. [BH-sbbCFA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation that is stateless-beyond-buffering. For an adversary $\mathcal{A}$ and a bit b, define the experiment $\mathbf{Exp}^{\text{ind-sbbcfa-b}}_{\mathcal{SE}}(\mathcal{A})$ as depicted in Fig. 9. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is given access to a special left-or-right encryption oracle LoR$(\cdot)$: on input two vectors

```
$\underline{\mathbf{Exp}^{\text{bh-sbbcfa-b}}_{\mathcal{SE}}(\mathcal{A})}$

    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, F \leftarrow \varepsilon, \mathsf{C} \leftarrow ()$
    $b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot),\mathsf{Dec}(\cdot)}$
    return $b'$

$\underline{\mathsf{LoR}((\mathbf{m}_0, \mathbf{m}_1))}$

    $\sigma_0 \leftarrow \sigma, \sigma_1 \leftarrow \sigma$
    $(\mathbf{c}_0, \sigma_0) \leftarrow \mathcal{E}_K(\mathbf{m}_0, \sigma_0)$
    $(\mathbf{c}_1, \sigma_1) \leftarrow \mathcal{E}_K(\mathbf{m}_1, \sigma_1)$
    $c_0 \leftarrow ||(\mathbf{c}_0), c_1 \leftarrow ||(\mathbf{c}_1)$
    if $|c_0| \neq |c_1|$ then return $\lightning$
    $\sigma \leftarrow \sigma_b$
    for $k = 1$ to $|\mathbf{c}_b|$
        $i \leftarrow i + 1$
        $\mathsf{C}_i \leftarrow \mathbf{c}_b(k)$
    return $c_b$

$\underline{\mathsf{Dec}(f)}$

    $M \leftarrow \varepsilon, \mathsf{prefix} \leftarrow 0$
    $len \leftarrow |f|$
    for $k = 1$ to $len$
        $(m, \varrho) \leftarrow \mathcal{D}_K(f[k], \varrho)$
        $m' \leftarrow m' \parallel m$
        $F \leftarrow F \parallel f[k]$
        if $\varrho = \varepsilon$ and $m'[|m'|] = \P$ then
            if $F \in \mathsf{C}$ then $m' \leftarrow \varepsilon$
            $M \leftarrow M \parallel m'$
            $F \leftarrow \varepsilon, m' \leftarrow \varepsilon$
    for all $c \in \mathsf{C}$
        if $F \diamond c = F$ then $\mathsf{prefix} \leftarrow 1$
    if $\mathsf{prefix} = 0$ then
        $M \leftarrow M \parallel m', m' \leftarrow \varepsilon$
    return $M$
```

Figure 9: Experiment to define BH-sbbCFA security.

of messages, either the left or the right result is returned, but with the caveat that the concatenation of ciphertexts is returned *only* if it has the same length in both worlds (but note that we do not insist that the two vectors of messages contain the same number of components). The adversary is additionally given a decryption oracle $\mathsf{Dec}(\cdot)$. It can query the decryption oracle on any sequence of ciphertext fragments, except that the output corresponding to a ciphertext previously output by the left-or-right oracle will be artificially suppressed.

The adversary's goal is to output a bit b', as its guess of the challenge bit b, and the experiment returns b' as well. We define the advantage of an adversary $\mathcal{A}$ as:

$$\mathbf{Adv}^{\text{ind-sbbcfa}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{ind-sbbcfa-1}}_{\mathcal{SE}}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-sbbcfa-0}}_{\mathcal{SE}}(\mathcal{A}) = 1\right].$$

The scheme $\mathcal{SE}$ is said to be IND-sbbCFA secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{ind-sbbcfa}}_{\mathcal{SE}}(\mathcal{A})$ is small.

It turns out that the above definition, which we argue is the natural analogue of the boundary-hiding definition in the stateful setting, is unsatisfiable by any 'reasonable' SBB encryption scheme, see the note at the end of this section. As such our coverage of boundary hiding in the SBB setting will be somewhat limited.

Relating Boundary Hiding to Indistinguishability. We now establish a few relations between notions of boundary hiding and notions of indistinguishability which we will use in later Sections. Theorem 5.3 states that for length-regular[2] schemes boundary hiding implies left-or-right indistinguishability. Intuitively this

[2]An encryption scheme is said to be *length-regular* if for all $m_1, m_2 \in \mathcal{M}$ where $|m_1| = |m_2|$ it holds (with probability 1) that $|\mathsf{Enc}_K(m_1)| = |\mathsf{Enc}_K(m_2)|$.

follows because the special left-or-right oracle in the BH-ATK notions can be used to simulate the left-or-right oracle in the IND-ATK notions. The requirement on length-regularity ensures that a valid query to the IND-ATK oracle results in a valid query to the BH-ATK oracle. Other than that the proof is straightforward and we omit it. Moreover, it is not too hard to show that BH-ATK is strictly stronger than IND-ATK: take any IND-ATK secure scheme and append each ciphertext with a special marker string, e.g., 1^{128}.

Theorem 5.3. [BH-ATK $\longrightarrow$ IND-ATK] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a length-regular symmetric encryption scheme supporting fragmentation. For any ATK $\in \{\mathsf{CPA}, \mathsf{sbbCFA}, \mathsf{sfCFA}\}$ and any IND-ATK adversary $\mathcal{A}_{ind}$ there exists a BH-ATK adversary $\mathcal{A}_{bh}$ consuming similar resources to $\mathcal{A}_{ind}$ such that:

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathcal{SE}}(\mathcal{A}_{ind}) \leq \mathbf{Adv}^{\text{bh-atk}}_{\mathcal{SE}}(\mathcal{A}_{bh}) .$$

Intuitively the concatenation of multiple random strings is indistinguishable from a single random string of the same length. It then follows that schemes having ciphertexts indistinguishable from random strings should also hide ciphertext boundaries. This is stated more formally, for the passive setting[3], in the following theorem. Again it is not hard to show that this implication is strict: take any BH-CPA secure scheme and re-encode its ciphertexts by doubling every bit, i.e., $0 \to 00$ and $1 \to 11$.

Theorem 5.4. [IND\$-CPA $\longrightarrow$ BH-CPA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme supporting fragmentation. For any BH-CPA adversary $\mathcal{A}_{bh}$ there exists an IND\$-CPA adversary $\mathcal{A}_{ind\$}$ consuming similar resources to $\mathcal{A}_{bh}$ such that:

$$\mathbf{Adv}^{\text{bh-cpa}}_{\mathcal{SE}}(\mathcal{A}_{bh}) \leq 2 \cdot \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}_{ind\$}) .$$

Proof. For any adversary $\mathcal{A}_{bh}$ we construct adversary $\mathcal{A}_{ind\$}$ as follows. Adversary $\mathcal{A}_{ind\$}$ picks a bit d uniformly at random and then runs $\mathcal{A}_{bh}$. Then $\mathcal{A}_{ind\$}$ uses this bit and its own encryption oracle to simulate the special left-or-right encryption oracle to $\mathcal{A}_{bh}$. That is, it uses d to pick the message vector, it encrypts each message in the vector componentwise, and returns their concatenation. If $\mathcal{A}_{bh}$'s output is equal to d, then $\mathcal{A}_{ind\$}$ outputs 1 else it outputs 0. Now when $\mathcal{A}_{ind\$}$ is run in the IND\$-CPA experiment with $b = 1$ it provides $\mathcal{A}_{bh}$ with a perfect simulation of the BH-CPA experiment with random bit d. Otherwise if $b = 0$ the responses to $\mathcal{A}_{bh}$'s queries are completely independent to the bit d because they are random strings of appropriate length. We thus have that:

$$\begin{aligned}
\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}_{ind\$}) &= \Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-1}}_{\mathcal{SE}}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-0}}_{\mathcal{SE}}(\mathcal{A}_{ind\$}) = 1\,\right] \\
&= \Pr\left[\,d \leftarrow \{0,1\} : \mathbf{Exp}^{\text{bh-cpa-d}}_{\mathcal{SE}}(\mathcal{A}_{bh}) = d\,\right] - \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{bh-cpa}}_{\mathcal{SE}}(\mathcal{A}_{bh}) - \frac{1}{2} \\
&= \frac{1}{2} \cdot \mathbf{Adv}^{\text{bh-cpa}}_{\mathcal{SE}}(\mathcal{A}_{bh}) .
\end{aligned}$$

□

[3] It can be shown that this implication does not hold in the stateful setting, in fact the stateful InterMAC construction of Section 7.2 serves as a separating example.

6 Denial of Service

In this section we study fragmentation-related Denial-of-Service (DoS) attacks. This is, to the best of our knowledge, the first formal treatment of DoS prevention as a property of a symmetric encryption scheme. In Section 1.1 we outlined such a DoS attack for the case of SSH. In that example, by carefully tampering with only a few bits in one of the transmitted ciphertexts, the adversary manages to 'confuse' the decryption algorithm so that it will produce no output until a huge amount of ciphertext is received. Informally this kind of attack is what our security notions will attempt to capture. We stress that such attacks are not specific to SSH, but relate more generally to schemes supporting fragmentation. We will equip the adversary with an encryption oracle and a decryption oracle. Its goal will be to produce a sequence of ciphertext fragments whose concatenation is at least n bits long, where each of these fragments decrypts to the empty string. We will then quantify the DoS security of a scheme via the minimum value of n such that no 'efficient' adversary is successful in producing such a sequence of fragments.

The countermeasure adopted by SSH to mitigate against such attacks cf. [20, Section 6.1] is to limit the maximum ciphertext length to n bits; thereby ensuring that the decryption algorithm will produce an output after at most n bits of ciphertext. In the case of OpenSSH n is set to 2^{21}. We consider this to be a serious limitation since it affects the usability of the scheme. If two parties wish to exchange large files, it is understood that this may require waiting for large amounts of ciphertext before recovering it at the receiver side, and this should be allowed. What we wish to avoid is cases where the communicating parties are exchanging short messages, but the adversary is able to tamper with the ciphertexts in such a way that the receiver has to wait for a large amount of ciphertext before producing an output. Thus we aim to formulate DoS security in a way that allows lowering n without necessarily restricting the maximum message size in the message space. To accommodate this we exclude trivial win conditions of the type where a passive adversary forwards ciphertexts (or fragmentations thereof) of length n or higher from the encryption oracle to the decryption oracle. In essence we will insist that the sequence of fragments by which the adversary wins be generated by an active adversary. In the stateful setting this means that we will require the 'winning' sequence of fragments to occur after the adversary has become active. In the SBB setting, trivial win conditions will be trickier to catch. We now formulate the two definitions more precisely.

6.1 Security Definitions

Definition 6.1. [n-DOS-sfCFA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a positive integer n, define the experiment $\mathbf{Exp}^{n\text{-dos-sfcfa}}_{\mathcal{SE}}(\mathcal{A})$ as depicted in Fig. 10. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$, and a stateful decryption oracle $\mathsf{sfDec}(\cdot)$. The adversary's goal is to submit to the stateful decryption oracle $\mathsf{sfDec}(\cdot)$ an out-of-sync sequence of fragments whose combined length is at least n bits, such that all fragments return no output upon decryption. In the event that the adversary succeeds, the experiment returns 1, and 0 otherwise. The output of the stateful decryption oracle is never suppressed.

We define the advantage of an adversary $\mathcal{A}$ as:

$$\mathbf{Adv}^{n\text{-dos-sfcfa}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{n\text{-dos-sfcfa}}_{\mathcal{SE}}(\mathcal{A}) = 1\right].$$

The scheme $\mathcal{SE}$ is said to be n-$\mathsf{DOS\text{-}sfCFA}$ secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{n\text{-dos-sfcca}}_{\mathcal{SE}}(\mathcal{A})$ is small.

```
$\mathbf{Exp}^{n\text{-dos-sfcfa}}_{\mathcal{SE}}(\mathcal{A})$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $C \leftarrow \varepsilon, F \leftarrow \varepsilon, \mathtt{C} \leftarrow ()$
    $i \leftarrow 0, j \leftarrow 0$
    $\mathsf{sync} \leftarrow 1, \mathsf{win} \leftarrow 0$
    $\mathcal{A}^{\mathsf{Enc}(\cdot), \mathsf{sfDec}(\cdot)}$
    return $\mathsf{win}$

$\mathsf{Enc}(m)$
    $(c, \sigma) \leftarrow \mathcal{E}_K(m, \sigma)$
    $i \leftarrow i + 1, \mathtt{C}_i \leftarrow c$
    return $c$

$\mathsf{sfDec}(f)$
    $(m, \varrho) \leftarrow \mathcal{D}_K(f, \varrho)$
    if $\mathsf{sync} = 1$ then
        $F \leftarrow F \parallel f$
        while $C \diamond F = C$ and $j < i$
            $j \leftarrow j + 1$
            $C \leftarrow C \parallel \mathtt{C}_j$
        if $F \diamond C \neq F$ then
            $\mathsf{sync} \leftarrow 0$
            if $m = \varepsilon$ then $F \leftarrow F \,\%\, C$
            else $F \leftarrow \varepsilon$
    else
        if $m = \varepsilon$ then $F \leftarrow F \parallel f$
        else $F \leftarrow \varepsilon$
    if $\mathsf{sync} = 0$ and $|F| \geq n$ then $\mathsf{win} \leftarrow 1$
    return $m$
```

Figure 10: Experiment to define n-DOS-sfCFA security.

The initial lines of code in the decryption oracle work as before: their purpose is to detect when the queries become out of sync, i.e. when the adversary becomes active. Once the queries have become out of sync, the variable F is used to store the last concatenation of out-of-sync fragments that did not return any output upon decryption. If at any point the size of F exceeds n, the win flag is set. In the case where the first out-of-sync fragment returns no output upon decryption, only the out-of-sync portion of that fragment is stored in F. That is, we measure the ciphertext from the point at which the tampering has occurred. This excludes trivial win conditions, resulting say from a legitimately-produced long ciphertext (longer than n bits) where the *last* bit is flipped by the adversary. Permitting such win cases would also require limiting a scheme's maximum message size for it to be secure.

We now define an analogous DoS security notion in the SBB setting. As before we want to exclude win conditions where the adversary merely forwards (possibly fragmented) ciphertexts from the encryption oracle to the decryption oracle. While in the stateful setting the adversary is considered active if he reorders or replays ciphertext, in the SBB setting we will consider this behaviour to be passive. Thus adversarial strategies that exploit reorderings and replays are deemed invalid in the SBB setting. This distinction between the stateful and SBB settings is present in all security notions considered thus far. Adapting this ideology to DoS security, if the winning sequence of fragments coincides with the start of a new ciphertext, we do not want it to correspond to a fragmentation of a long ciphertext that was previously output by the encryption oracle. Moreover, it should not be prefixed by a previously-output ciphertext, or by a prefix of a previously-output ciphertext. A sequence of fragments that satisfies these requirements is said to be *non-trivial*.

Definition 6.2. [n-DOS-sbbCFA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a positive integer n, define the experiment $\mathbf{Exp}^{n\text{-dos-sfcfa}}_{\mathcal{SE}}(\mathcal{A})$ as depicted in Fig. 11. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$, and a decryption oracle $\mathsf{Dec}(\cdot)$. The adversary's goal is to

```
$\mathbf{Exp}^{n\text{-dos-sbbcfa}}_{\mathcal{SE}}(\mathcal{A})$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $F \leftarrow \varepsilon, \mathsf{C} \leftarrow ()$
    $i \leftarrow 0, q \leftarrow 0$
    $\mathsf{tmp} \leftarrow 0, \mathsf{win} \leftarrow 0$
    $\mathcal{A}^{\mathsf{Enc}(\cdot),\mathsf{Dec}(\cdot)}$
    return $\mathsf{win}$

$\mathsf{Enc}(m)$
    $(c, \sigma) \leftarrow \mathcal{E}_K(m, \sigma)$
    $i \leftarrow i + 1, \mathsf{C}_i \leftarrow c$
    return $c$

$\mathsf{Dec}(f)$
    $M \leftarrow \varepsilon, m' \leftarrow \varepsilon$
    $len \leftarrow |f|$
    for $k = 1$ to $len$
        $(m, \varrho) \leftarrow \mathcal{D}_K(f[k], \varrho)$
        $m' \leftarrow m' \parallel m$
        $F \leftarrow F \parallel f[k]$
        if $\varrho = \varepsilon$ and $m'[|m'|] = \P$ then
            $M \leftarrow M \parallel m'$
            $F \leftarrow \varepsilon, m' \leftarrow \varepsilon$
    if $M \parallel m' \neq \varepsilon$ then $q \leftarrow |F|$
    if $|F| - q \geq n$ then
        $\mathsf{tmp} \leftarrow 1$
        for all $c \in \mathsf{C}$
            if $|F \,\%\, c| < n$ then $\mathsf{tmp} \leftarrow 0$
        $\mathsf{win} \leftarrow \mathsf{tmp}$
    return $M \parallel m'$
```

Figure 11: Experiment to define n-DOS-sbbCFA security.

submit to the decryption oracle $\mathsf{Dec}(\cdot)$ a non-trivial sequence of fragments whose combined length is at least n bits, such that all fragments return no output upon decryption. In the event that the adversary succeeds, the experiment returns 1, and 0 otherwise. The output of the decryption oracle is never suppressed.

We define the advantage of an adversary $\mathcal{A}$ in this experiment as:

$$\mathbf{Adv}^{n\text{-dos-sbbcfa}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\, \mathbf{Exp}^{n\text{-dos-sbbcfa}}_{\mathcal{SE}}(\mathcal{A}) = 1 \,\right].$$

The scheme $\mathcal{SE}$ is said to be n-DOS-sbbCFA secure if, for every adversary $\mathcal{A}$ with reasonable resources, its advantage $\mathbf{Adv}^{n\text{-dos-sbbcfa}}_{\mathcal{SE}}(\mathcal{A})$ is small.

Once again we exploit literal decryption to decrypt ciphertext fragments incrementally. This allows the decryption oracle to detect ciphertext boundaries in order to filter out trivial win conditions. The variable F stores the concatenation of all ciphertext bits belonging to the current ciphertext. If the end of ciphertext is detected (by checking for the condition where $\varrho = \varepsilon$ **and** $m'[|m'|] = \P$), then F is reset. The variable q points to the end of the last received fragment within F that produced an output. Each time a fragment is received the decryption oracle checks whether the concatenation of fragments that did not produce an output, i.e. $F[q + 1, |F|]$, is at least n bits long. If so it further verifies that after all possible replayed ciphertext prefixes are removed, it still is at least n bits long.

A Note on DoS and Ciphertext Integrity. In an attempt to limit our scope we did not formulate a notion of ciphertext integrity for schemes supporting fragmentation. Nonetheless, we wish to emphasise that DoS security does not imply ciphertext integrity, nor the other way round. While a notion of ciphertext integrity would ensure that an adversarially generated ciphertext is never accepted, it does not guarantee at which point it will be rejected. Thus, as in the case of SSH, it may be that a ciphertext can only be rejected once the (possibly very large) ciphertext has been received in full. On the other hand, if a scheme is n-DOS-sfCFA secure it does not guarantee that adversarially generated ciphertexts will be rejected. We purposefully chose to maintain this separation between the two notions, as we feel that the two security goals are rather different. This said, a combination of the two security notions has practical significance, since it guarantees that any tampering in the communication would be detected within n bits. Intuitively it is easy to see that the InterMAC constructions, presented in the next section, achieve this combined security goal.

7 Constructions

7.1 Applying Instantaneously Decodable Postprocessing (IDP)

We now present a simple transformation for converting a symmetric encryption scheme to an encryption scheme that supports ciphertext fragmentation. In addition we will see that if the scheme that we start with is IND-sfCCA secure, then the constructed scheme will be IND-sfCFA secure. Similarly if we start with a scheme that is IND-CCA secure, the constructed scheme will satisfy IND-sbbCFA security. The construction will make use of an instantaneously decodable encoding scheme. Later we will see that if we allow the encoding scheme to be keyed and probabilistic, the construction can in addition achieve boundary hiding against passive adversaries. Towards this goal, we extend the syntax of encoding schemes as follows.

Generalised Encoding Schemes. An encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ is a triple of algorithms with an associated word space $\mathcal{W} \subseteq \{0,1\}^*$. The randomised *key-generation* algorithm $\mathcal{K}_c$ takes no input and returns a secret key K. The *encoding* algorithm $\mathcal{EC}$ which may be stateful, probabilistic, or both, takes as input a secret key K and a string (word) $w \in \mathcal{W}$ to return a codeword $v \in \{0,1\}^*$. The deterministic *decoding* algorithm $\mathcal{DC}$ takes as input a secret key K and a codeword $v \in \{0,1\}^*$ to return a word $w \in \mathcal{W} \cup \{\varepsilon\}$, possibly followed by other outputs. For any key K, we denote the range of the encoding algorithm by $\mathcal{EC}_K(\mathcal{W})$.

Definition 7.1. [Instantaneous Decodability] An encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ with associated wordspace $\mathcal{W} \subseteq \{0,1\}^*$, is said to be instantaneously decodable if for all keys K that can be output by $\mathcal{K}_c$, it holds that:

1. For all $w \in \mathcal{W}$, and all $s \in \{0,1\}^*$, if $v \leftarrow \mathcal{EC}_K(w)$ then $(w, s) \leftarrow \mathcal{DC}_K(v \parallel s)$.
2. For all $s \in \{0,1\}^*$, if no $v \in \mathcal{EC}_K(\mathcal{W})$ is a prefix of s then $(\varepsilon, s) \leftarrow \mathcal{DC}_K(s)$.

Note that instantaneous decodability does not require the encoding scheme to be keyed. In fact any keyless encoding scheme that is prefix-free is also instantaneously decodable. Later in this section we will give an example of a keyed encoding scheme that is instantaneously decodable. Throughout we assume that $\mathcal{K}_c$, $\mathcal{EC}$, and $\mathcal{DC}$ are efficiently computable algorithms.

Construction 7.2. [The IDP Construction] Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with associated message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ be an instantaneously decodable encoding scheme with an associated word space that contains $\mathcal{C}$. Then the construction specified in Figure 12 yields an encryption scheme supporting fragmentation $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ with an associated message space $\mathcal{M}$. Furthermore if $\mathcal{SE}$ is stateless, then $\overline{\mathcal{SE}}$ is stateless beyond buffering.

Correctness of the constructed scheme $\overline{\mathcal{SE}}$ follows immediately from the correctness of $\mathcal{SE}$ and the instantaneous decodability of $\mathcal{ES}$. Furthermore if $\mathcal{SE}$'s decryption algorithm is stateless, the only state that $\overline{\mathcal{D}}$

```
Algorithm $\overline{\mathcal{K}}$
    $K_c \leftarrow \mathcal{K}_c$
    $(K_e, \sigma, \varrho) \leftarrow \mathcal{K}_e$
    $K \leftarrow K_c \parallel K_e$
    return $(K, \sigma, (\varrho, \varepsilon))$

Algorithm $\overline{\mathcal{E}}_K(m, \sigma)$
    $(c, \sigma) \leftarrow \mathcal{E}_{K_e}(m, \sigma)$
    $v \leftarrow \mathcal{EC}_{K_c}(c)$
    return $(v, \sigma)$

Algorithm $\overline{\mathcal{D}}_K(f, (\varrho, \alpha))$
    $m' \leftarrow \varepsilon, w \leftarrow f$
    $\alpha \leftarrow \alpha \parallel f$
    while $(w \neq \varepsilon)$
        $(w, \alpha) \leftarrow \mathcal{DC}_{K_c}(\alpha)$
        if $(w \neq \varepsilon)$ then
            $(m, \varrho) \leftarrow \mathcal{D}_{K_e}(w, \varrho)$
            $m' \leftarrow m' \parallel m \parallel ¶$
    return $(m', (\varrho, \alpha))$
```

Figure 12: The constructed scheme $\overline{\mathcal{SE}}$ using instantaneously decodable postpocessing.

maintains is the buffer α. Now the buffer is always initialised to ε, and it is easy to see that after decrypting any complete ciphertext, the buffer will always be empty. Finally, because all submitted fragments are appended to the buffer from which ciphertexts are then extracted and submitted to $\mathcal{D}$, decryption is independent of the fragmentation pattern. Hence the scheme also satisfies literal decryption, and consequently is stateless beyond buffering.

Note that by instantiating the encoding scheme with a prefix-free encoding, we get a very efficient transformation for converting a 'standard' symmetric encryption scheme to an encryption scheme that supports fragmentation. We next show the nice property that if we start from a scheme that is IND-sfCCA secure, the transformation yields a scheme that is IND-sfCFA secure.

Theorem 7.3. [IDP is IND-sfCFA secure] Let $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the scheme from Construction 7.2, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and an instantaneously decodable encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$. Then for any IND-sfCFA adversary $\mathcal{A}_{sfcfa}$ against $\overline{\mathcal{SE}}$, there exists an IND-sfCCA adversary $\mathcal{A}_{sfcca}$ against $\mathcal{SE}$ such that:

$$\mathbf{Adv}^{\text{ind-sfcfa}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{sfcfa}) \leq \mathbf{Adv}^{\text{ind-sfcca}}_{\mathcal{SE}}(\mathcal{A}_{sfcca}), \qquad (1)$$

where $\mathcal{A}_{sfcca}$ consumes similar resources to $\mathcal{A}_{sfcfa}$.

Proof. For any adversary $\mathcal{A}_{sfcfa}$ we construct adversary $\mathcal{A}_{sfcca}$ as follows. Adversary $\mathcal{A}_{sfcca}$ runs $\mathcal{K}_c$ to generate an encoding key and then runs $\mathcal{A}_{sfcfa}$. It then uses the encoding key together with its left-or-right oracle to simulate $\mathcal{A}_{sfcfa}$'s left-or-right oracle as per Construction 7.2, keeping record of the ciphertexts it returns. $\mathcal{A}_{sfcfa}$'s decryption queries are handled by maintaining a buffer to which the queried fragments are appended. Then $\mathcal{A}_{sfcca}$ repeatedly applies the decoding algorithm to the buffer until no codeword can be extracted, and submits the codewords, in the same order, to its own stateful decryption oracle. When the queries become out of sync, the returned messages are appended with ¶, concatenated together, and the resulting string is returned to $\mathcal{A}_{sfcfa}$. $\mathcal{A}_{sfcca}$ uses its records to keep track of when $\mathcal{A}_{sfcfa}$'s queries become out of sync. This is necessary since the first out of sync query might correspond to an encryption of ε, and $\mathcal{A}_{sfcfa}$ would not be able to distinguish this case from a message being suppressed because it is in sync. Finally $\mathcal{A}_{sfcca}$ outputs whatever $\mathcal{A}_{sfcfa}$ outputs.

```
$\underline{\mathbf{Exp}^{\text{rpe-b}}_{\mathcal{ES}}(\mathcal{A})}$
    $K \leftarrow \mathcal{K}_c$
    $b' \leftarrow \mathcal{A}^{\mathsf{EoR}(\cdot)}$
    return $b'$

$\underline{\mathsf{EoR}(\ell)}$
    $w \leftarrow_{\$} \{0,1\}^{\ell}$
    $v \leftarrow \mathcal{EC}_K(w)$
    if $b = 0$ then
        $v \leftarrow_{\$} \{0,1\}^{|v|}$
    return $v$
```

Figure 13: Experiment to define randomness preserving encodings.

From the instantaneous decodability of $\mathcal{ES}$ it follows that $\mathcal{A}_{sfcca}$'s decryption queries will be in sync if and only if $\mathcal{A}_{sfcfa}$'s decryption queries are in sync. Therefore $\mathcal{A}_{sfcca}$ provides $\mathcal{A}_{sfcfa}$ with a perfect simulation of its environment. Thus:

$$\Pr\left[d \leftarrow_{\$} \{0,1\} \,:\, \mathbf{Exp}^{\text{ind-sfcfa-d}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{sfcfa}) = d \right] \leq \Pr\left[b \leftarrow_{\$} \{0,1\} \,:\, \mathbf{Exp}^{\text{ind-sfcca-b}}_{\mathcal{SE}}(\mathcal{A}_{sfcca}) = b \right],$$

and equation (1) follows. □

The following analogous theorem is implied by a similar proof which we omit to avoid repetition.

Theorem 7.4. [IDP is IND-sbbCFA secure] Let $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the scheme from Construction 7.2, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ with stateless decryption, and an instantaneously decodable encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$. Then for any IND-sbbCFA adversary $\mathcal{A}_{sbbcfa}$ against $\overline{\mathcal{SE}}$, there exists an IND-CCA adversary $\mathcal{A}_{cca}$ against $\mathcal{SE}$ such that:

$$\mathbf{Adv}^{\text{ind-sbbcfa}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{sbbcfa}) \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(\mathcal{A}_{cca}),$$

where $\mathcal{A}_{cca}$ consumes similar resources to $\mathcal{A}_{sbbcfa}$.

Construction 7.2 shows that IND-sfCFA and IND-sbbCFA security are not hard to attain. However when we instantiate the encoding scheme with a prefix free encoding, ciphertext boundaries will inevitably be revealed. While this is what allows the constructed scheme to support ciphertext fragmentation, it obviously conflicts with the goal of boundary hiding. We partly solve this conflict by employing a keyed encoding scheme, which reveals ciphertext boundaries solely to the holder of the encoding key. We now formulate a security property for encoding schemes that will allow Construction 7.2 to achieve IND\$-CPA security, and by Theorem 5.4 hide ciphertext boundaries from passive adversaries.

Definition 7.5. [Randomness Preserving Encodings] Let $\mathcal{L}$ be a non-empty set of positive integers, and let $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ be an encoding scheme with associated word space $\mathcal{W} = \bigcup_{\ell \in \mathcal{L}} \{0,1\}^{\ell}$. For an adversary $\mathcal{A}$ and a bit b define the experiment $\mathbf{Exp}^{\text{rpe-b}}_{\mathcal{ES}}(\mathcal{A})$ as shown in Figure 13. The experiment starts by calling $\mathcal{K}_c$ to generate an encoding key K_c. The adversary $\mathcal{A}$ is then given access to an encode-or-random oracle $\mathsf{EoR}(\cdot)$, that it can query on any length value $\ell \in \mathcal{L}$. Depending on the value of b, the oracle will either return an encoding of a random string of length ℓ, or a random string of the same length as that encoding.

The adversary's goal is to output a bit b', as its guess of the challenge bit b, and the experiment returns b' as well. We define the adversary's rpe-advantage as:

$$\mathbf{Adv}^{\mathrm{rpe}}_{\mathcal{ES}}(\mathcal{A}) = \Pr\left[\,\mathbf{Exp}^{\mathrm{rpe}\text{-}1}_{\mathcal{ES}}(\mathcal{A}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\mathrm{rpe}\text{-}0}_{\mathcal{ES}}(\mathcal{A}) = 1\,\right] .$$

The encoding scheme $\mathcal{ES}$ is said to be a *randomness preserving encoding* (RPE) scheme, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\mathrm{rpe}}_{\mathcal{ES}}(\mathcal{A})$ is small.

Theorem 7.6. [IDP is IND\$-CPA secure] Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with associated message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ be an encoding scheme with an associated word space that contains $\mathcal{C}$. Define the encryption scheme supporting fragmentation $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ according to Construction 7.2. For any IND\$-CPA adversary $\mathcal{A}_{ind\$}$ against $\overline{\mathcal{SE}}$, there exist adversaries $\mathcal{A}'_{ind\$}$ and $\mathcal{A}_{rpe}$ such that:

$$\mathbf{Adv}^{\mathrm{ind\$\text{-}cpa}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) \leq \mathbf{Adv}^{\mathrm{ind\$\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}'_{ind\$}) + \mathbf{Adv}^{\mathrm{rpe}}_{\mathcal{ES}}(\mathcal{A}_{rpe}) , \tag{2}$$

where $\mathcal{A}'_{ind\$}$ and $\mathcal{A}_{rpe}$ consume similar resources to $\mathcal{A}_{ind\$}$.

Proof. To prove Theorem 7.6 we introduce we introduce a hybrid experiment $\mathbf{ExpH}$, similar in spirit to the two IND\$-CPA experiments corresponding to each bit value. The hybrid experiment proceeds exactly as $\mathbf{Exp}^{\mathrm{ind\$\text{-}cpa\text{-}1}}_{\overline{\mathcal{SE}}}$, except that the encryption oracle returns encodings of random strings instead. More specifically, after computing an encryption under $\mathcal{SE}$ of the queried message, it picks uniformly at random a string of the same length as the ciphertext, and returns an encoding under $\mathcal{ES}$ of this string instead. We then have that:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind\$\text{-}cpa}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) = &\Big(\Pr\left[\,\mathbf{Exp}^{\mathrm{ind\$\text{-}cpa\text{-}1}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right]\Big) \\ &+ \Big(\Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\mathrm{ind\$\text{-}cpa\text{-}0}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) = 1\,\right]\Big) . \end{aligned} \tag{3}$$

Now we consider each of the above terms in the braces separately. For any adversary $\mathcal{A}_{ind\$}$ distinguishing between the two experiments in the first term, we construct an IND\$-CPA adversary $\mathcal{A}'_{ind\$}$ against $\mathcal{SE}$. Adversary $\mathcal{A}'_{ind\$}$ runs $\mathcal{K}_c$ to obtain an encoding key and then runs $\mathcal{A}_{ind\$}$. It then simulates $\mathcal{A}_{ind\$}$'s encryption oracle in accordance with the IDP construction, except that it uses its own oracle to compute encryptions under $\mathcal{SE}$. It then outputs whatever $\mathcal{A}_{ind\$}$ outputs. Note that when $\mathcal{A}'_{ind\$}$'s oracle returns real ciphertexts, it provides $\mathcal{A}_{ind\$}$ with a perfect simulation of the IND\$-CPA experiment with a bit value of one. Alternatively when $\mathcal{A}'_{ind\$}$'s oracle returns random strings, it provides $\mathcal{A}_{ind\$}$ with a perfect simulation of the hybrid experiment. Hence:

$$\Pr\left[\,\mathbf{Exp}^{\mathrm{ind\$\text{-}cpa\text{-}1}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right] \leq \mathbf{Adv}^{\mathrm{ind\$\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}'_{ind\$}) \tag{4}$$

Similarly, for any adversary $\mathcal{A}_{ind\$}$ distinguishing between the two experiments in the second term, we construct an RPE adversary $\mathcal{A}_{rpe}$ against $\mathcal{ES}$. Adversary $\mathcal{A}_{rpe}$ runs $\mathcal{K}_e$ to obtain an encryption key and then runs $\mathcal{A}_{ind\$}$. It simulates $\mathcal{A}_{ind\$}$'s encryption oracle by computing an encryption of the queried message under $\mathcal{SE}$, it then queries its own oracle with the length of this ciphertext and forwards the response to $\mathcal{A}_{ind\$}$. It then outputs whatever $\mathcal{A}'_{cpa}$ outputs. When $\mathcal{A}_{rpe}$'s oracle returns encodings of random strings, it provides $\mathcal{A}_{ind\$}$

```
Algorithm $\mathcal{K}_c$
    $K \leftarrow_{\$} \mathcal{K}$
    return $K$

Algorithm $\mathcal{EC}_K(w)$
    $x \leftarrow_{\$} \{0,1\}^n$
    $y \leftarrow \langle |w| \rangle_l \oplus F_K(x)$
    $v \leftarrow x \parallel y \parallel w$
    return $v$

Algorithm $\mathcal{DC}_K(v)$
    if $|v| \leq n + l$ then
        return $(\varepsilon, v)$
    $len \leftarrow F_K(v[1,n]) \oplus v[n+1,l]$
    if $|v| - n - l < len$ then
        return $(\varepsilon, v)$
    $w \leftarrow v[n+l+1, n+l+len]$
    $z \leftarrow v[n+l+len+1, |v|]$
    return $(w, z)$
```

Figure 14: The encoding scheme of Theorem 7.7 that is both instantaneously decodable and randomness preserving.

with a perfect simulation of the hybrid experiment. Otherwise when $\mathcal{A}_{rpe}$'s oracle returns random strings, it provides $\mathcal{A}_{ind\$}$ with a perfect simulation of the IND\$-CPA experiment with a bit value of zero. Therefore:

$$\Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-0}}_{\mathcal{SE}}(\mathcal{A}_{ind\$}) = 1\,\right] \leq \mathbf{Adv}^{\text{rpe}}_{\mathcal{ES}}(\mathcal{A}_{rpe}) \tag{5}$$

Combining equations (3),(4), and (5) yields equation (2), as desired. □

We now complete the IDP construction by showing a simple instantiation of an encoding scheme that is both instantaneously decodable and randomness preserving. The encoding scheme is constructed from a pseudorandom function family mapping n bit strings to l bit strings, and is presented in Figure 14.

Theorem 7.7. [IDP Instantiation] Let $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^l$ be a function family indexed by the set $\mathcal{K}$. Then Figure 14 defines an instantaneously decodable encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ with word space $\mathcal{W} = \bigcup_{\ell \leq l}\{0,1\}^{\ell}$. Moreover, for any RPE adversary $\mathcal{A}_{rpe}$ against $\mathcal{ES}$ making at most q queries, there exists a PRF adversary $\mathcal{A}_{prf}$ such that:

$$\mathbf{Adv}^{\text{rpe}}_{\mathcal{ES}}(\mathcal{A}_{rpe}) \leq \mathbf{Adv}^{\text{prf}}_{F}(\mathcal{A}_{prf}) + \left(\frac{q^2}{2^{n+1}}\right), \tag{6}$$

where adversary $\mathcal{A}_{prf}$ consumes similar resources to $\mathcal{A}_{rpe}$.

Proof. We first outline why the encoding scheme is instantaneously decodable. Note that the decoding algorithm first recovers the length field and then uses this value to determine where the codeword ends. Thus the first requirement of Definition 7.1 is satisfied. As for the second requirement, note that the only case where the input string s is not prefixed by a valid code word is either when its length is less than or equal to $n + l$, or the recovered length value is greater than the length of the remaining string. In both cases the decoding algorithm returns (ε, s), as required.

We now prove that the encoding scheme is randomness preserving. To do this, we show that from any RPE adversary $\mathcal{A}_{rpe}$, we can build a PRF adversary $\mathcal{A}_{prf}$ against F. Adversary $\mathcal{A}_{prf}$ runs $\mathcal{A}_{rpe}$, and simulates its oracle by sampling random strings of the queried length and encoding them according to the construction of Figure 14 and computing PRF values using its own oracle. $\mathcal{A}_{prf}$ keeps a record of all the

n bit strings that it samples, and if at any point a collision occurs it outputs 0 and halts. Otherwise $\mathcal{A}_{prf}$ outputs whatever $\mathcal{A}_{rpe}$ outputs. Note that when $\mathcal{A}_{prf}$ is instantiated with F, it responds to $\mathcal{A}_{rpe}$'s queries with real encodings of random strings. On the other hand if its oracle is a random function it returns uniformly random strings (since it never queries its oracle on the same value more than once). Let Z^b represent the event $\mathbf{Exp}_{\mathcal{ES}}^{\text{rpe-b}}(\mathcal{A}_{rpe}) = 1$, and let E represent the event that a collision occurs when sampling n bit strings. Then we have that:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{ES}}^{\text{rpe}}(\mathcal{A}_{rpe}) &= \Pr\left[\, Z^1 \wedge \overline{E} \,\right] - \Pr\left[\, Z^0 \wedge \overline{E} \,\right] + \left(\Pr\left[\, Z^1 \wedge E \,\right] - \Pr\left[\, Z^0 \wedge E \,\right]\right) \\ &\leq \Pr\left[\, Z^1 \wedge \overline{E} \,\right] - \Pr\left[\, Z^0 \wedge \overline{E} \,\right] + \Pr\left[\, E \,\right].\end{aligned}$$

Applying a birthday bound to E, and substituting for the other terms we get:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{ES}}^{\text{rpe}}(\mathcal{A}_{rpe}) \leq \Pr&\left[\, K \leftarrow_{\$} \mathcal{K} : \mathcal{A}_{prf}^{F_K(\cdot)} = 1 \,\right] \\ &- \Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(n, l) : \mathcal{A}_{prf}^{f(\cdot)} = 1 \,\right] + \left(\frac{q^2}{2^{n+1}}\right). \qquad (7)\end{aligned}$$

Equation (6) then follows from equation (7). □

The IDP construction is attractive in terms of efficiency, modularity, and versatility. If we look at prior constructions, we see that achieving confidentiality and hiding boundaries while supporting fragmentation was already a source of conflict. Consider SSH for instance. Its effort to encrypt the length field can be interpreted as an attempt to hide boundaries. When instantiated with CBC encryption, it is easy to see that SSH achieves BH-CPA security, but as evidenced by the attack from [1] it is insecure in the IND-sfCFA sense. Alternatively if we look at TLS, the result of [15] implies that it is IND-sfCCA secure. Moreover the length field contained in the header works as a prefix free encoding, and therefore by Theorem 7.3, TLS is IND-sfCFA secure. However since the header is in cleartext the scheme obviously does not achieve BH-CPA security.

7.2 The InterMAC Construction

We now move to a more ambitious goal, to simultaneously achieve all three of our security notions. In comparison to the IDP construction and SSH, we now additionally consider boundary hiding against active adversaries and DoS security. None of the schemes considered thus far achieve boundary hiding in the active setting. To see the difficulty with this consider once more the case of SSH. Given a concatenation of ciphertexs, the adversary can now flip the first bit and submit it bit by bit to its decryption oracle until an error is returned, which marks the first ciphertext boundary. In addition to achieving boundary hiding security in the active setting, the scheme that we present in this section also achieves N-DOS-sfCFA security *without* limiting the maximum message size to N bits.

Our proposed scheme breaks a message into equal-sized segments and encrypts them separately. It then appends a MAC tag to each intermediate ciphertext and concatenates them to produce the final ciphertext. The sender and receiver keep a state which contains a message and a segment number to be used in the MAC computation. Each segment uses a bit flag to indicate the last segment in a message. We now describe the construction in more detail.

```
Algorithm $\overline{\mathcal{K}}$
    $(K_e, \sigma_e, \varrho_e) \leftarrow \mathcal{K}_e$
    $K_m \leftarrow \mathcal{K}_m$
    $K \leftarrow K_e \parallel K_m$
    $\sigma \leftarrow (\sigma_e, 0)$
    $\varrho \leftarrow (\varrho_e, \varepsilon, \varepsilon, 0, 0, 0)$
    return $(K, \sigma, \varrho)$

Algorithm $\overline{\mathcal{E}}_K(m, (\sigma_e, i))$
    $c \leftarrow \varepsilon, b \leftarrow 0, i \leftarrow i + 1$
    for $j = 1$ to $|m|/\ell_m$
        $p \leftarrow 1 + (j-1).\ell_m$
        $q \leftarrow j.\ell_m$
        $m' \leftarrow m[p, q]$
        if $q = |m|$ then $b \leftarrow 1$
        $(c', \sigma_e) \leftarrow \mathcal{E}_{K_e}(b \parallel m', \sigma_e)$
        $\tau \leftarrow \mathcal{T}_{K_m}(\langle i \rangle \parallel \langle j \rangle \parallel c')$
        $c \leftarrow c \parallel c' \parallel \tau$
    return $(c, (\sigma_e, i))$

Algorithm $\overline{\mathcal{D}}_K(f, (\varrho_e, \alpha, m, i, j, \mathsf{fail}))$
    $w \leftarrow \varepsilon, \alpha \leftarrow \alpha \parallel f$
    while $|\alpha| \geq N$
        $c \leftarrow \alpha[1, \ell_c], \tau \leftarrow \alpha[\ell_c + 1, N]$
        $\alpha \leftarrow \alpha[N + 1, |\alpha|]$
        $j \leftarrow j + 1$
        $v \leftarrow \mathcal{V}_{K_m}(\langle i \rangle \parallel \langle j \rangle \parallel c, \tau)$
        if $v = \perp$ and $\mathsf{fail} = 0$ then
            $w \leftarrow w \parallel \perp, \mathsf{fail} \leftarrow 1$
        else if $\mathsf{fail} = 1$ then
            $w \leftarrow w \parallel \perp$
        else
            $(m', \varrho_e) \leftarrow \mathcal{D}_{K_e}(c, \varrho_e)$
            $m \leftarrow m \parallel m'[2, \ell_m + 1]$
            if $m'[1] = 1$ then
                $w \leftarrow w \parallel m \parallel ¶$
                $i \leftarrow i + 1, j \leftarrow 0, m \leftarrow \varepsilon$
    return $(w, (\varrho_e, \alpha, m, i, j, \mathsf{fail}))$
```

Figure 15: The stateful InterMAC construction $\mathcal{IM}$.

Construction 7.8. [InterMAC] Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme such that its message space contains $\{0,1\}^{\ell_m+1}$, for some desired $\ell_m \in \mathbb{N}$. Furthermore let $\mathcal{E}$ be length-regular, such that it maps all messages of length ℓ_m to ciphertexts of length ℓ_c. Let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a message authentication code with associated tag length ℓ_{tag} and message space $\{0,1\}^*$. Then the stateful InterMAC construction, specified in Figure 15, yields an encryption scheme supporting fragmentation $\mathcal{IM} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ with message space $\{\{0,1\}^{\ell_m}\}^+$. The ciphertext segment size N associated to the stateful InterMAC construction is given by $N = \ell_c + \ell_{tag}$.

At first sight Figure 15 may seem daunting. Accordingly we now give an informal description. Each message is split into chunks of ℓ_m bits, a bit is then prepended to each chunk and encrypted separately. For all chunks of plaintext except the last, the prepended bit is set to zero. For each of these ciphertexts c', a MAC tag is computed over the concatenation of the encoded message counter $\langle i \rangle$, the encoded segment index $\langle j \rangle$, and the ciphertext. These ciphertext-tag pairs are then concatenated to yield the final ciphertext. Decryption starts by appending the input ciphertext fragment f to the buffer string α, and resetting the output plaintext string w. The while loop then extracts ciphertext segments from the buffer one at a time. Each segment is parsed into a ciphertext and a MAC tag, and the tag is then verified. The returned output string w is then constructed as follows. For valid ciphertexts, i.e. ciphertexts where all segments contain a valid tag, the plaintext is only returned when the last ciphertext segment has been received. Alternatively, once an invalid segment is encountered, the $\perp$ symbol is returned for that segment and every segment (irrespective of its validity) that is received thereafter.

Theorem 7.9. [InterMAC is N-DOS-sfCFA secure] Let $\mathcal{IM} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the InterMAC scheme from

Construction 7.8, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Let its segment size be N. Then for any N-DOS-sfCFA adversary $\mathcal{A}_{dos}$ against $\mathcal{IM}$, there exists a UF-CMA adversary $\mathcal{A}_{uf}$ against $\mathcal{MA}$ such that:

$$\mathbf{Adv}_{\mathcal{IM}}^{N\text{-dos-sfcfa}}(\mathcal{A}_{dos}) \leq \mathbf{Adv}_{\mathcal{MA}}^{\text{uf-cma}}(\mathcal{A}_{uf}),$$

where $\mathcal{A}_{uf}$ consumes similar resources to $\mathcal{A}_{dos}$.

Proof. Consider the $\mathbf{Exp}_{\mathcal{IM}}^{n\text{-dos-sfcfa}}(\mathcal{A}_{dos})$ experiment for $n = N$. At any point in time, let F^* be the concatenation of all ciphertext fragments queried by $\mathcal{A}_{dos}$, and let u be the largest non-negative integer such that the substring $F^*[1, uN]$ is in sync. Let E represent the event that $|F^*| \geq (u+1)N$ and sfDec$(\cdot)$ did not return any output after receiving the $((u+1)N)^{\text{th}}$ bit. For the case of InterMAC, if the adversary wins the experiment then E must have occurred. We now bound the probability of event E occurring.

Adversary $\mathcal{A}_{uf}$ runs $\mathcal{K}_e$ to get an encryption key and initialise the states. It then runs $\mathcal{A}_{dos}$ and uses the encryption key together with its tagging oracle to simulate $\mathcal{A}_{dos}$'s encryption oracle (as per Construction 7.8). In addition it maintains an ordered list of all the ciphertexts it returns, together with their corresponding messages. $\mathcal{A}_{dos}$'s decryption queries are then handled as follows. $\mathcal{A}_{uf}$ maintains the string F^* (as defined above), and uses it together with the other list to keep track of when $\mathcal{A}_{dos}$ becomes active. Moreover it maintains the decryption counters i and j (as per Construction 7.8). While $\mathcal{A}_{dos}$'s queries are in sync, it uses its list to simulate the decryption oracle. When it happens that $|F^*| \geq (u+1)N$, it parses $F^*[uN+1, (u+1)N]$ into a ciphertext c and a MAC tag τ, submits the pair $(\langle i \rangle \parallel \langle j \rangle \parallel c, \tau)$ to its verification oracle, and halts.

Note that until $|F^*| \geq (u+1)N$ happens, $\mathcal{A}_{uf}$'s simulation of $\mathcal{A}_{dos}$'s environment is perfect. Moreover, counters i and j ensure that the only possible time where $\mathcal{A}_{uf}$ queried a string with these values is when it computed the tag for the j^{th} segment of the i^{th} ciphertext (if such a segment existed). However, since $\mathcal{MA}$ is a MAC and by assumption $F^*[uN+1, (u+1)N]$ does not match that segment, it must be that the corresponding ciphertext components do not match either. It thus follows that whenever E occurs, $\mathcal{A}_{uf}$ produces a valid MAC forgery and wins the UF-CMA experiment. We then have that:

$$\Pr\left[\, \mathbf{Exp}_{\mathcal{IM}}^{N\text{-dos-sfcfa}}(\mathcal{A}_{dos})) = 1 \,\right] \leq \Pr[\, E \,] \leq \Pr\left[\, \mathbf{Exp}_{\mathcal{MA}}^{\text{uf-cma}}(\mathcal{A}_{uf}) = 1 \,\right],$$

and equation (7.9) thus follows. □

Note that we only have BH-sfCFA security left to prove, since IND-sfCFA security will then be implied by Theorem 5.3.

Theorem 7.10. [InterMAC is BH-sfCFA secure] Let $\mathcal{IM} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the InterMAC scheme from Construction 7.8, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Then for any BH-sfCFA adversary $\mathcal{A}_{sfcfa}$ against $\mathcal{IM}$, there exists adversaries $\mathcal{A}_{cpa}$, $\mathcal{A}_{prf}$, and $\mathcal{A}_{uf}$ such that:

$$\frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{IM}}^{\text{bh-sfcfa}}(\mathcal{A}_{sfcfa}) \leq \mathbf{Adv}_{\mathcal{SE}}^{\text{ind\$-cpa}}(\mathcal{A}_{cpa}) + \mathbf{Adv}_{\mathcal{T}}^{\text{prf}}(\mathcal{A}_{prf}) + \mathbf{Adv}_{\mathcal{MA}}^{\text{uf-cma}}(\mathcal{A}_{uf}), \tag{8}$$

where all three adversaries consume similar resources to $\mathcal{A}_{sfcfa}$.

```
$\underline{\mathbf{ExpA}^{\mathrm{b}}_{\mathcal{SE}}(\mathcal{A})}$
  $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
  $j \leftarrow 1, \mathsf{sync} \leftarrow 1$
  $C \leftarrow \varepsilon, F \leftarrow \varepsilon$
  $b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot),\mathsf{sfDec}(\cdot)}$
  return $b'$

$\underline{\mathsf{LoR}((\mathbf{m}_0, \mathbf{m}_1))}$
  $\sigma_0 \leftarrow \sigma, \sigma_1 \leftarrow \sigma$
  $(\mathbf{c}_0, \sigma_0) \leftarrow \mathcal{E}_K(\mathbf{m}_0, \sigma_0)$
  $(\mathbf{c}_1, \sigma_1) \leftarrow \mathcal{E}_K(\mathbf{m}_1, \sigma_1)$
  $c_0 \leftarrow ||(\mathbf{c}_0), c_1 \leftarrow ||(\mathbf{c}_1)$
  if $|c_0| \neq |c_1|$ then return $\lightning$
  $\sigma \leftarrow \sigma_b, C \leftarrow C \parallel c_b$
  return $c_b$

$\underline{\mathsf{sfDec}(f)}$
  $m \leftarrow \varepsilon, F \leftarrow F \parallel f$
  while $|F| - jN \geq 0$
    $p \leftarrow 1 + (j-1)N, q \leftarrow jN$
    if $\mathsf{sync} = 1$ then
      if $F[p, q] \neq C[p, q]$ then
        $\mathsf{sync} \leftarrow 0, m \leftarrow \perp$
    else
      $m \leftarrow m \parallel \perp$
    $j \leftarrow j + 1$
  return $m$
```

Figure 16: The auxiliary experiment used to prove Theorem 7.10.

Proof. We will prove Theorem 7.10 in two parts. For the first part of the proof we will make use of the auxiliary experiment $\mathbf{ExpA}^{\mathrm{b}}_{\mathcal{IM}}$ of Figure 16. This is essentially the $\mathbf{Exp}^{\text{bh-sfcfa-b}}_{\mathcal{IM}}$ experiment with a modified stateful decryption oracle. Now the stateful decryption oracle does not return any output until the queries become out of sync, at which point it returns $\perp$ at every N-bit boundary of ciphertext that it receives. At any point in time, let F^* be the concatenation of all ciphertext fragments queried by $\mathcal{A}_{sfcfa}$, and let u be the largest non-negative integer such that the substring $F^*[1, uN]$ is in sync. Let E represent the event that in the BH-sfCFA experiment $|F^*| \geq (u+1)N$ and $\mathsf{sfDec}(\cdot)$ did not return $\perp$ after receiving the first $((u+1)N)$ bits. Let W denote the event $\mathbf{Exp}^{\text{bh-sfcfa-b}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) = b$ and let W^A denote the event $\mathbf{ExpA}^{\mathrm{d}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) = d$, where bits b and d are picked uniformly at random. We thus have that:

$$\Pr\left[\, W \,\right] - \Pr\left[\, W^A \,\right] = \Pr\left[\, W \wedge E \,\right] + \Pr\left[\, W \wedge \overline{E} \,\right] - \Pr\left[\, W^A \,\right] .$$

Due to the details of the InterMAC construction, the two experiments are identical if E does not occur. Bounding $\Pr\left[\, \overline{E} \,\right]$, cancelling equal terms, and then bounding $\Pr\left[\, W \wedge E \,\right]$ yields:

$$\begin{aligned} \Pr\left[\, W \,\right] - \Pr\left[\, W^A \,\right] &\leq \Pr\left[\, W \wedge E \,\right] + \Pr\left[\, W \mid \overline{E} \,\right] - \Pr\left[\, W^A \,\right] \\ &\leq \Pr\left[\, W \wedge E \,\right] \\ &\leq \Pr\left[\, E \,\right] . \end{aligned} \tag{9}$$

Using a reduction similar to that in the proof of Theorem 7.9, it follows that there exists a UF-CMA adversary $\mathcal{A}_{uf}$ such that:

$$\Pr\left[\, E \,\right] \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) . \tag{10}$$

Combining equations (9) and (10), and then multiplying by two and subtracting one on each side of the inequality, yields:

$$\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{\mathit{sfcfa}}) \leq \big(2 \cdot \Pr\left[\, W^A \,\right] - 1\big) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{\mathit{uf}})\,. \tag{11}$$

Now from any adversary $\mathcal{A}_{\mathit{sfcfa}}$, we can construct a BH-CPA adversary $\mathcal{A}''_{\mathit{cpa}}$ against $\mathcal{IM}$ as follows. $\mathcal{A}''_{\mathit{cpa}}$ runs $\mathcal{A}_{\mathit{sfcfa}}$, and forwards its encryption queries to its own encryption oracle while keeping record of all ciphertexts that it returns. Decryption queries are handled by running the $\mathsf{sfDec}(\cdot)$ algorithm of Figure 16. Finally $\mathcal{A}''_{\mathit{cpa}}$ outputs whatever $\mathcal{A}_{\mathit{sfcfa}}$ outputs. Note that $\mathcal{A}''_{\mathit{cpa}}$ provides $\mathcal{A}_{\mathit{sfcfa}}$ with a perfect simulation of the auxiliary experiment. It then follows that:

$$\Pr\left[\, W^A \,\right] = \Pr\left[\, d \leftarrow_{\$} \{0,1\} \,:\, \mathbf{Exp}^{\text{bh-cpa-d}}_{\mathcal{IM}}(\mathcal{A}''_{\mathit{cpa}}) = d \,\right]\,. \tag{12}$$

Combining equations (11) and (12), we obtain:

$$\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{\mathit{sfcfa}}) \leq \mathbf{Adv}^{\text{bh-cpa}}_{\mathcal{IM}}(\mathcal{A}''_{\mathit{cpa}}) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{\mathit{uf}})\,, \tag{13}$$

and then applying Theorem 5.4 yields:

$$\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{\mathit{sfcfa}}) \leq 2 \cdot \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{IM}}(\mathcal{A}'_{\mathit{cpa}}) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{\mathit{uf}})\,. \tag{14}$$

We now move to the second part of the proof and bound the advantage of $\mathcal{A}'_{\mathit{cpa}}$. Towards this aim we introduce a hybrid experiment $\mathbf{ExpH}$, similar in spirit to the two IND\$-CPA experiments corresponding to each bit value. The hybrid experiment proceeds exactly as $\mathbf{Exp}^{\text{ind\$-cpa-1}}_{\mathcal{IM}}$, except for one detail. In the encryption oracle, for every ciphertext segment, the MAC tag is replaced with a uniformly random string of length ℓ_{tag}. Then we have that:

$$\begin{aligned}\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{IM}}(\mathcal{A}'_{\mathit{cpa}}) = &\Big(\Pr\left[\, \mathbf{Exp}^{\text{ind\$-cpa-1}}_{\mathcal{IM}}(\mathcal{A}'_{\mathit{cpa}}) = 1 \,\right] - \Pr\left[\, \mathbf{ExpH}(\mathcal{A}'_{\mathit{cpa}}) = 1 \,\right]\Big) \\ &+ \Big(\Pr\left[\, \mathbf{ExpH}(\mathcal{A}'_{\mathit{cpa}}) = 1 \,\right] - \Pr\left[\, \mathbf{Exp}^{\text{ind\$-cpa-0}}_{\mathcal{IM}}(\mathcal{A}'_{\mathit{cpa}}) = 1 \,\right]\Big)\,. \end{aligned} \tag{15}$$

Now consider each of the above terms in the braces separately. For any adversary $\mathcal{A}'_{\mathit{cpa}}$ distinguishing between the two experiments in the first term, we can associate a PRF adversary $\mathcal{A}_{\mathit{prf}}$ against $\mathcal{T}$. Adversary $\mathcal{A}_{\mathit{prf}}$ runs $\mathcal{K}_e$ to obtain an encryption key and initialise the states, and then runs $\mathcal{A}'_{\mathit{cpa}}$. It simulates its encryption oracle in accordance with the InterMAC scheme, except that it uses its own oracle to compute the MAC tags. It then outputs whatever $\mathcal{A}'_{\mathit{cpa}}$ outputs. Note that if $\mathcal{A}_{\mathit{prf}}$'s oracle is instantiated with $\mathcal{T}$, it perfectly simulates a 'real' encryption oracle for $\mathcal{A}'_{\mathit{cpa}}$. On the other hand if its oracle is a random function it simulates the encryption oracle of the hybrid experiment, as long as it does not query the random function on the same input more than once. The counters in the InterMAC construction guarantee that this never occurs. Therefore:

$$\Pr\left[\, \mathbf{Exp}^{\text{ind\$-cpa-1}}_{\mathcal{IM}}(\mathcal{A}'_{\mathit{cpa}}) = 1 \,\right] - \Pr\left[\, \mathbf{ExpH}(\mathcal{A}'_{\mathit{cpa}}) = 1 \,\right] \leq \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{\mathit{prf}})\,. \tag{16}$$

Similarly for any adversary $\mathcal{A}'_{\mathit{cpa}}$ distinguishing between the two experiments in the second term we construct an IND\$-CPA adversary $\mathcal{A}_{\mathit{cpa}}$ against $\mathcal{SE}$. Adversary $\mathcal{A}_{\mathit{cpa}}$ runs $\mathcal{A}'_{\mathit{cpa}}$ simulating its encryption oracle in accordance with the InterMAC scheme, except that it uses its own oracle to compute encryptions under $\mathcal{SE}$, and replaces tag values with random strings of length ℓ_{tag}. It then outputs whatever $\mathcal{A}'_{\mathit{cpa}}$ outputs. When $\mathcal{A}_{\mathit{cpa}}$'s oracle returns real ciphertexts, it provides $\mathcal{A}'_{\mathit{cpa}}$ with a perfect simulation of the hybrid experiment.

```
Algorithm $\overline{\mathcal{K}}$
  $(K_e, \sigma, \varepsilon) \leftarrow \mathcal{K}_e$
  $K_m \leftarrow \mathcal{K}_m$
  $K \leftarrow K_e \parallel K_m$
  $\varrho \leftarrow (\varepsilon, \varepsilon, 0)$
  return $(K, \sigma, \varrho)$

Algorithm $\overline{\mathcal{E}}_K(m, \sigma)$
  $c \leftarrow \varepsilon, \tau \leftarrow 0^{\ell_{tag}}, b \leftarrow 0$
  for $j = 1$ to $|m|/\ell_m$
    $p \leftarrow 1 + (j-1).\ell_m$
    $q \leftarrow j.\ell_m$
    $m' \leftarrow m[p, q]$
    if $q = |m|$ then $b \leftarrow 1$
    $(c', \sigma) \leftarrow \mathcal{E}_{K_e}(b \parallel m', \sigma)$
    $\tau \leftarrow \mathcal{T}_{K_m}(\tau \parallel c')$
    $c \leftarrow c \parallel c' \parallel \tau$
  return $(c, \sigma)$

Algorithm $\overline{\mathcal{D}}_K(f, (\alpha, m, \tau_0))$
  $w \leftarrow \varepsilon, \alpha \leftarrow \alpha \parallel f$
  while $|\alpha| \geq N$
    $c \leftarrow \alpha[1, \ell_c], \tau \leftarrow \alpha[\ell_c + 1, N]$
    $\alpha \leftarrow \alpha[N + 1, |\alpha|]$
    $v \leftarrow \mathcal{V}_{K_m}(\tau_0 \parallel c, \tau)$
    $\tau_0 \leftarrow \tau$
    if $v = \perp$ and $m \neq \oslash$ then
      $w \leftarrow w \parallel \perp, m \leftarrow \oslash$
    else if $m = \oslash$ then
      $w \leftarrow w \parallel \perp$
    else
      $(m', \varrho_e) \leftarrow \mathcal{D}_{K_e}(c, \varepsilon)$
      $m \leftarrow m \parallel m'[2, \ell_m + 1]$
      if $m'[1] = 1$ then
        $w \leftarrow w \parallel m \parallel ¶$
        $\tau_0 \leftarrow 0^{\ell_{tag}}, m \leftarrow \varepsilon$
  return $(w, (\alpha, m, \tau_0))$
```

Figure 17: The stateless beyond buffering InterMAC construction $\mathcal{IM}^*$.

Alternatively when $\mathcal{A}_{cpa}$'s oracle returns random strings, it provides $\mathcal{A}'_{cpa}$ with a perfect simulation of the IND\$-CPA experiment with a bit value of zero. Hence:

$$\Pr\left[\,\mathbf{ExpH}(\mathcal{A}'_{cpa}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-0}}_{\mathcal{IM}}(\mathcal{A}'_{cpa}) = 1\,\right] \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa})\,. \tag{17}$$

Combining equations (14),(15),(16), and (17) yields (8), as desired. □

7.3 A SBB Variant of InterMAC

Construction 7.11. [SBB InterMAC] Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with stateless decryption, having a message space containing $\{0,1\}^{\ell_m+1}$ for some desired $\ell_m \in \mathbb{N}$, and error space $\mathcal{S}_\perp$. Furthermore let $\mathcal{E}$ be length-regular, such that it maps all messages of length ℓ_m to ciphertexts of length ℓ_c. Let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a message authentication code with associated tag length ℓ_{tag} and message space $\{0,1\}^*$. Let $\oslash$ be such that $\oslash \notin \mathcal{S}_\perp$. Then the SBB InterMAC construction, specified in Figure 17, yields an SBB encryption scheme supporting fragmentation $\mathcal{IM}^* = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ with message space $\{\{0,1\}^{\ell_m}\}^+$. The ciphertext segment size N associated to the stateful InterMAC construction is given by $N = \ell_c + \ell_{tag}$.

The above construction works analogously to its stateful counterpart, with a few exceptions. Counters are no longer maintained, and are therefore not included in the MAC tag computation and verification. Instead, the tag of the previous segment is prepended to the ciphertext when computing the MAC tag. For the purpose of computing the tag in the first segment of each ciphertext, the previous tag value is set to $0^{\ell_{tag}}$. In decryption, the fail flag has been dropped, and we now set m to the special symbol $\oslash$ instead.

Thus as before, once an invalid MAC tag is detected, the decryption algorithm always returns $\perp$ from that point onwards. Note that this does not violate the SBB definition, see Section 3.2. Finally, the construction assumes an invertible encoding mapping the triple (α, m, τ_0) to a single string, such that $(\varepsilon, \varepsilon, 0^{\ell_{tag}})$ is mapped to the empty string. This technicality is required for the scheme to satisfy the SBB definition. Note that the decryption state does not contain more information than a buffer storing all bits pertaining to the ciphertext being decrypted. In fact it would have been functionally equivalent to let the decryption state be such a buffer, flushed only when the end of a ciphertext is found, and compute (α, m, τ_0) from this buffer each time the decryption algorithm is invoked. However we chose this implementation since it is less wasteful in computational resources, and yet satisfies the SBB definition.

Theorem 7.12. [SBB InterMAC is N-DOS-sbbCFA secure] Let $\mathcal{IM}^* = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the SBB InterMAC scheme from Construction 7.11, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ with stateless decryption and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Let its segment size be N. Then for any N-DOS-sbbCFA adversary $\mathcal{A}_{dos}$ against $\mathcal{IM}^*$ whose encryption queries total at most μ_e bits, there exist adversaries $\mathcal{A}_{uf}$ and $\mathcal{A}_{prf}$ such that:

$$\mathbf{Adv}^{N\text{-dos-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{dos}) \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}}}\right), \tag{18}$$

where $\mathcal{A}_{uf}$ and $\mathcal{A}_{prf}$ consume similar resources to $\mathcal{A}_{dos}$.

Proof. Consider the experiment $\mathbf{Exp}^{N\text{-dos-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{dos})$. Let F and $\mathtt{C}$ be as in Figure 11, and let u be the largest non-negative integer such that there exists a $c \in \mathtt{C}$ satisfying $F[1, uN] \diamond c = F[1, uN]$. Let E represent the event that $|F| \geq (u+1)N$ and $\mathsf{Dec}(\cdot)$ did not return any output after receiving the $((u+1)N)^{\text{th}}$ bit. For the case of InterMAC, if the adversary wins the experiment, then E must have occurred. Furthermore, let Q represent the event that for any two ciphertexts c and c' returned by the encryption oracle before E has occurred, there exist positive integers x and y, where $x \leq y$, such that $c[xN - \ell_{tag} + 1, xN] = c'[yN - \ell_{tag} + 1, yN]$ but $c[1, xN] \neq c'[1, yN]$, or $c[xN - \ell_{tag} + 1, xN] = 0^{\ell_{tag}}$. Thus Q represents the event that either two tags collide or a tag value of all zeros occurs. We then have that:

$$\begin{aligned}\Pr\left[\, \mathbf{Exp}^{N\text{-dos-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{dos})) = 1 \,\right] &= \Pr[\, E \,] = \Pr[\, E \wedge Q \,] + \Pr\left[\, E \wedge \overline{Q} \,\right], \\ &\leq \Pr[\, Q \,] + \Pr\left[\, E \mid \overline{Q} \,\right]. \end{aligned} \tag{19}$$

We now bound the probability of event Q occurring. Towards this goal we construct from $\mathcal{A}_{dos}$ a PRF adversary $\mathcal{A}_{prf}$ against $\mathcal{T}$. It starts by running $\mathcal{K}_e$ to get an encryption key and initialise the states. It then runs $\mathcal{A}_{dos}$ and uses the encryption key together with its oracle to simulate $\mathcal{A}_{dos}$'s encryption oracle (as per Construction 7.11). In addition it maintains a list of all the ciphertexts it returns, together with their corresponding messages. Note that by assumption event E has not occurred yet, thus $\mathcal{A}_{prf}$ is able to simulate $\mathcal{A}_{dos}$'s decryption oracle by using this list. Therefore, when $\mathcal{A}_{prf}$'s oracle is instantiated with $\mathcal{T}$ it provides $\mathcal{A}_{dos}$ with a perfect simulation of its environment. $\mathcal{A}_{prf}$ runs until $\mathcal{A}_{dos}$ halts or E occurs, at which point it checks whether Q has occurred. If so it outputs 1 otherwise it outputs 0. $\mathcal{A}_{prf}$ can check for Q as it proceeds by maintaining a list of the strings which it queried to its oracle, indexed by the returned tag values, and check for collisions or tag values of $0^{\ell_{tag}}$ while it populates the list. Consider now the case where $\mathcal{A}_{prf}$'s oracle is a random function. The probability of a collision in the tags can be bounded using a standard birthday bound, while the probability of a tag value of $0^{\ell_{tag}}$ is given by the number of queries divided by $2^{\ell_{tag}}$. Applying the union bound on these two probabilities, we have that:

$$\Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(\ell_c + \ell_{tag}, \ell_{tag}) : \mathcal{A}_{prf}^{f(\cdot)} = 1 \,\right] \leq \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}+1}}\right) + \left(\frac{\mu_e}{\ell_m \cdot 2^{\ell_{tag}}}\right).$$

Rounding the above bound, and applying it to $\mathcal{A}_{prf}$'s advantage formula, yields:

$$\Pr[\,Q\,] \le \mathbf{Adv}^{\mathrm{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}}}\right). \tag{20}$$

We now bound the second term of inequality (19), by constructing a UF-CMA adversary $\mathcal{A}_{uf}$ against $\mathcal{MA}$ from $\mathcal{A}_{dos}$. Adversary $\mathcal{A}_{uf}$ proceeds similarly to $\mathcal{A}_{prf}$. It runs $\mathcal{K}_e$ and uses this key together with its tagging oracle to simulate $\mathcal{A}_{dos}$'s encryption oracle. It also maintains a list of all the ciphertexts it returns together with their corresponding messages, and uses this to simulate $\mathcal{A}_{dos}$'s decryption oracle. It then keeps on simulating $\mathcal{A}_{dos}$'s environment until it halts or $|F| \ge (u+1)N$. If $|F| \ge (u+1)N$ happens with $u > 0$, it submits the pair $(F[uN - \ell_{tag} + 1, uN + \ell_c], F[uN + \ell_c + 1, (u+1)N])$ to its verification oracle, and halts. Alternatively, if $|F| \ge (u+1)N$ occurs with $u = 0$, it submits the pair $(0^{\ell_{tag}} \parallel F[1, \ell_c], F[\ell_c + 1, N])$ instead. Note that until it occurs that $|F| \ge (u+1)N$, $\mathcal{A}_{uf}$'s simulation of $\mathcal{A}_{dos}$'s environment is perfect. Furthermore, if Q did not occur, it follows that the first component of the submitted pair was not previously queried to the tagging oracle. Thus assuming Q did not occur, whenever E occurs, $\mathcal{A}_{uf}$'s submitted pair constitutes a valid forgery. Therefore:

$$\Pr\left[\,E \mid \overline{Q}\,\right] \le \mathbf{Adv}^{\mathrm{uf\text{-}cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}). \tag{21}$$

Combining equations (19),(20) and (21), yields (18), as desired. □

A slightly different analysis could be used to achieve a possibly better bound for Theorem 7.12. In particular when bounding event Q, we could have considered the probability that both the tag and ciphertext values collide. This would lower the birthday bound term, at the expense of introducing an extra term of the form $\mathbf{Adv}^{\mathrm{ind\$\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A})$. If the birthday bound is the dominant term, such an approach would yield a tighter bound. However we opted for a simpler proof of security.

Theorem 7.13. [SBB InterMAC is IND-sbbCFA secure] Let $\mathcal{IM}^* = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the SBB InterMAC scheme from Construction 7.11, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ with stateless decryption and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Let its segment size be N. Then for any IND-sbbCFA adversary $\mathcal{A}_{sbbcfa}$ against $\mathcal{IM}^*$ whose encryption queries total at most μ_e bits, there exist adversaries $\mathcal{A}_{cpa}$, $\mathcal{A}_{prf}$ and $\mathcal{A}_{uf}$, such that:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind\text{-}sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) \le \mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) + 2 \cdot \mathbf{Adv}^{\mathrm{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) \\ + 2 \cdot \mathbf{Adv}^{\mathrm{uf\text{-}cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}-1}},\end{aligned} \tag{22}$$

where all four adversaries consume similar resources to $\mathcal{A}_{sbbcfa}$.

Proof. The proof of Theorem 7.13 follows the same lines as its stateful analogue. For the first part of the proof we will make use of the auxiliary experiment $\mathbf{ExpA}^{\mathrm{b}}_{\mathcal{IM}}$ of Figure 18. This is essentially the $\mathbf{Exp}^{\mathrm{ind\text{-}sbbcfa\text{-}b}}_{\mathcal{IM}^*}$ experiment, with the difference that once the decryption oracle detects a ciphertext which is not a replay of another ciphertext output by the encryption oracle, it then returns $\perp$ at every N-bit boundary of ciphertext that it receives. Now let F and C be as in Figure 7, and let u be the largest non-negative integer such that there exists a $c \in \mathsf{C}$ satisfying $F[1, uN] \diamond c = F[1, uN]$. Let E represent the event that in the IND-sbbCFA experiment $|F| \ge (u+1)N$ and Dec$(\cdot)$ did not return $\perp$ after receiving the first $((u+1)N)$ bits. Let

```
$\underline{\mathbf{ExpA}^{\mathrm{b}}_{\mathcal{SE}}(\mathcal{A})}$
    $(K, \sigma, \varepsilon) \leftarrow \mathcal{K}$
    $i \leftarrow 0, j \leftarrow 1$
    $\mathsf{fail} \leftarrow 0, p \leftarrow 1$
    $\mathtt{C} \leftarrow (), F \leftarrow \varepsilon$
    $b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot),\mathsf{Dec}(\cdot)}$
    return $b'$

$\underline{\mathsf{LoR}((m_0, m_1))}$
    if $|m_0| \neq |m_1|$ then return $\lightning$
    $(c, \sigma) \leftarrow \mathcal{E}_K(m_b, \sigma)$
    $i \leftarrow i + 1, \mathtt{C}_i \leftarrow c$
    return $c$

$\underline{\mathsf{Dec}(f)}$
    $m \leftarrow \varepsilon, F \leftarrow F \parallel f$
    while $|F| - jN \geq 0$
        if $\mathsf{fail} = 1$ then $m \leftarrow m \parallel \perp$
        else
            $\mathsf{match} \leftarrow 0$
            for all $c \in \mathtt{C}$
                if $F[p, jN] \diamond c = F[p, jN]$
                    then $\mathsf{match} \leftarrow 1$
                if $F[p, jN] = c$
                    then $p \leftarrow jN + 1$
            if $\mathsf{match} = 0$ then
                $m \leftarrow m \parallel \perp, \mathsf{fail} \leftarrow 1$
        $j \leftarrow j + 1$
    $F \leftarrow F[p, |F|], p \leftarrow 1, j \leftarrow \lfloor |F|/N \rfloor$
    return $m$
```

Figure 18: The auxiliary experiment used to prove Theorem 7.13.

W denote the event $\mathbf{Exp}^{\text{ind-sbbcfa-b}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) = b$ and let W^A denote the event $\mathbf{ExpA}^{\mathrm{d}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) = d$, where bits b and d are picked uniformly at random. We then have that:

$$\Pr[\,W\,] - \Pr\left[\,W^A\,\right] = \Pr[\,W \wedge E\,] + \Pr\left[\,W \wedge \overline{E}\,\right] - \Pr\left[\,W^A\,\right] .$$

Due to the details of the InterMAC construction, the two experiments are identical if E does not occur. Bounding $\Pr\left[\,\overline{E}\,\right]$, cancelling equal terms, and then bounding $\Pr[\,W \wedge E\,]$ yields:

$$\begin{aligned}\Pr[\,W\,] - \Pr\left[\,W^A\,\right] &\leq \Pr[\,W \wedge E\,] + \Pr\left[\,W \mid \overline{E}\,\right] - \Pr\left[\,W^A\,\right] \\ &\leq \Pr[\,W \wedge E\,] \leq \Pr[\,E\,] . \qquad (23)\end{aligned}$$

Using a reduction similar to that in the proof of Theorem 7.12, it follows that there exist adversaries $\mathcal{A}_{uf}$ and $\mathcal{A}_{prf}$ such that:

$$\Pr[\,E\,] \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}}}\right) . \qquad (24)$$

Combining equations (23) and (24), and manipulating terms, yields:

$$\Pr[\,W\,] \leq \Pr\left[\,W^A\,\right] + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}}}\right) .$$

Multiplying both sides by two and subtracting one:

$$\mathbf{Adv}^{\text{ind-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) \leq \left(2 \cdot \Pr\left[\, W^A \,\right] - 1\right) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf})\,.$$

$$+\, 2 \cdot \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}-1}}\right) \tag{25}$$

Now from any adversary $\mathcal{A}_{sbbcfa}$, we can construct an IND-CPA adversary $\mathcal{A}_{cpa}$ against $\mathcal{SE}$ as follows. Adversary $\mathcal{A}_{cpa}$ runs $\mathcal{K}_m$ to obtain a key for the MAC and then runs $\mathcal{A}_{sbbcfa}$. It simulates its encryption oracle in accordance with the InterMAC scheme, except that it uses its own oracle to compute encryptions under $\mathcal{SE}$. In addition it maintains a list of all ciphertexts that it returns. Decryption queries are handled by simulating the $\mathsf{Dec}(\cdot)$ oracle of Figure 18. Finally $\mathcal{A}_{cpa}$ outputs whatever $\mathcal{A}_{sbbcfa}$ outputs. Note that $\mathcal{A}_{cpa}$ provides $\mathcal{A}_{sbbcfa}$ with a perfect simulation of the auxiliary experiment. It then follows that:

$$\Pr\left[\, W^A \,\right] = \Pr\left[\, d \leftarrow_{\$} \{0,1\} \;:\; \mathbf{Exp}^{\text{ind-cpa-d}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) = d \,\right]. \tag{26}$$

Combining equations (25) and (26) yields (22), as desired.

□

References

[1] Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext recovery attacks against SSH. In *2009 IEEE Symposium on Security and Privacy*, pages 16–26. IEEE Computer Society Press, May 2009. 3, 4, 5, 16, 28

[2] Gregory V. Bard. A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL. In Manu Malek, Eduardo Fernández-Medina, and Javier Hernando, editors, *SECRYPT*, pages 99–109. INSTICC Press, 2006. 7

[3] Gregory V. Bard. Blockwise-adaptive chosen-plaintext attack and online modes of encryption. In Steven D. Galbraith, editor, *11th IMA International Conference on Cryptography and Coding*, volume 4887 of *LNCS*, pages 129–151. Springer, Heidelberg, December 2007. 7

[4] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Trans. Inf. Syst. Secur.*, 7(2):206–241, 2004. 3, 4, 5, 7, 12, 13, 14

[5] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. On symmetric encryption with distinguishable decryption failures. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *LNCS*, pages 367–390. Springer, Heidelberg, March 2013. 8, 10

[6] Alexandra Boldyreva and Nut Taesombut. Online encryption schemes: New security notions and constructions. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *LNCS*, pages 1–14. Springer, Heidelberg, February 2004. 7

[7] Jean Paul Degabriele and Kenneth G. Paterson. Attacking the IPsec standards in encryption-only configurations. In *2007 IEEE Symposium on Security and Privacy*, pages 335–349. IEEE Computer Society Press, May 2007. 16

[8] Jean Paul Degabriele and Kenneth G. Paterson. On the (in)security of IPsec in MAC-then-encrypt configurations. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 493–504. ACM Press, October 2010. 4, 16

[9] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE Symposium on Security and Privacy*, pages 332–346. IEEE Computer Society Press, May 2012. 16

[10] Pierre-Alain Fouque, Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Authenticated on-line encryption. In Mitsuru Matsui and Robert J. Zuccherato, editors, *SAC 2003: 10th Annual International Workshop on Selected Areas in Cryptography*, volume 3006 of *LNCS*, pages 145–159. Springer, Heidelberg, August 2003. 7

[11] Pierre-Alain Fouque, Antoine Joux, and Guillaume Poupard. Blockwise adversarial model for on-line ciphers and symmetric encryption schemes. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, volume 3357 of *LNCS*, pages 212–226. Springer, Heidelberg, August 2004. 7

[12] Pierre-Alain Fouque, Gwenaëlle Martinet, and Guillaume Poupard. Practical symmetric on-line encryption. In Thomas Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *LNCS*, pages 362–375. Springer, Heidelberg, February 2003. 7

[13] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. 40

[14] Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Blockwise-adaptive attackers: Revisiting the (in)security of some provably secure encryption models: CBC, GEM, IACBC. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *LNCS*, pages 17–30. Springer, Heidelberg, August 2002. 4, 7

[15] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 372–389. Springer, Heidelberg, December 2011. 5, 28

[16] Kenneth G. Paterson and Gaven J. Watson. Plaintext-dependent decryption: A formal security treatment of SSH-CTR. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 345–361. Springer, Heidelberg, May 2010. 3, 4, 6, 7, 39

[17] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Heidelberg, February 2004. 10

[18] Cihangir Tezcan and Serge Vaudenay. On hiding a plaintext length by preencryption. In Javier Lopez and Gene Tsudik, editors, *ACNS 11: 9th International Conference on Applied Cryptography and Network Security*, volume 6715 of *LNCS*, pages 345–358. Springer, Heidelberg, June 2011. 5

[19] Andrew M. White, Austin R. Matthews, Kevin Z. Snow, and Fabian Monrose. Phonotactic reconstruction of encrypted VoIP conversations: Hookt on fon-iks. In *2011 IEEE Symposium on Security and Privacy*, pages 3–18. IEEE Computer Society Press, May 2011. 16

[20] T. Ylonen and C. Lonvick. The secure shell (SSH) transport layer protocol. RFC 4253 (Proposed Standard), January 2006. Updated by RFC 6668. 20

A The Paterson–Watson Model

To analyse SSH in the context of fragmentation, Paterson and Watson already extended Bellare et al.'s IND-sfCCA notion to incorporate fragmentation. However, their security definition [16, Definition 2] is tailored specifically to work with SSH, to the extent that the security experiments are *directly* referring to quantities that are SSH specific. For instance, the sequence number and buffer *as used by SSH* are also crucial for the definition of security. This already makes their definition very difficult to work with in general.

Nonetheless, one could try to extract a security definition by abstracting away the various SSH specific quantities, or duplicating them explicitly in the experiment, e.g. let the experiment explicitly keep track of a sequence number (as we did for our IND-sfCFA experiment) and/or a buffer (as we did for our IND-sbbCFA game). This itself is not at all trivial, and it would surface the following three problems buried in the Paterson–Watson experiments:

1. Fragments containing multiple, complete ciphertexts are dealt with in a peculiar way, as the decryption oracle will only ever output one message at a time. This can cause a considerable lag, in the sense that submitting one fragment to the decryption oracle might actually result in a much older fragment being decrypted (and returned), whereas the fresh fragment is simply added to the buffer in full (to be processed later). This is illustrated in Fig. 19.

2. A correctness requirement is completely missing. One can try to adapt our correctness definition, but it does not seem to blend well with their choice to output only a single message at a time. (Weaker notions of correctness are also possible.)

3. A 'bombing' behaviour is enforced in the security experiment: once a decryption returns $\perp$, all subsequent decryptions will return $\perp$. This (needlessly) restricts the security definition to schemes that actually implement this behaviour in their decryption algorithm. Even if the security notion can only be met by schemes that employ this 'bombing' behaviour, it should not be part of the security experiment.

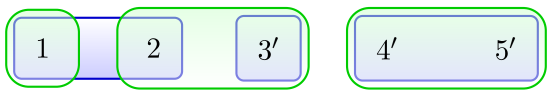

Figure 19: The Paterson–Watson game would (after buffering $f_1 = (1)$) consider the fragment $f_2 = (23')$ still in-sync, leading to (suppressed) decryption of ciphertext $c_1 = (1)$ only. The third fragment $f_3 = (4'5')$ triggers decryption of ciphertext $c_2 = (3')$ whose deviation (from 3) causes this third fragment to be out-of-sync. The actual contents of f_3 itself are irrelevant here.

B Unsatisfiability of SBB Boundary Hiding

We now outline a general attack, applicable to any practically relevant scheme, showing that the BH-sbbCFA definition given in Figure 9 is unsatisfiable. The reader is recommended to first refer to the next section where security against Denial of Service attacks is introduced and defined.

Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be any SBB encryption scheme supporting fragmentation that is n-DOS-sbbCFA secure for some value n. Furthermore, let m_1 and m_2 be any two messages such that $|\mathcal{E}_K(m_1)| < |\mathcal{E}_K(m_2)|$. An adversary can then query the message-vector pair $([m_1, m_2], [m_2, m_1])$ to the special left-or-right oracle and get a concatenation of ciphertexts c^*. It then chops off the last $|\mathcal{E}_K(m_2)|$ bits from c^* to get c', and submits the string $c' \,\|\, c' \,\|\, \ldots \,\|\, c'$ (possibly in fragments) to the decryption oracle. The number of copies of c' that are included in this string is such that its total length exceeds n. Now if c^* corresponds to the first message vector then c' will be a prohibited ciphertext and all output will be suppressed by the decryption oracle. On the other hand if c^* corresponds to the second message vector, then by the correctness of the scheme c' will not be a prohibited ciphertext and the concatenated string is guaranteed to produce some output by the n-DOS-sbbCFA security of the scheme. Thus the presence or absence of any output from the decryption oracle will indicate to the adversary which message vector was encrypted.

While our formulation of BH-sbbCFA is quite natural, one could argue that the reason it is unsatisfiable is because the set of prohibited ciphertexts $\mathtt{C}$ depends on the challenge bit b. A possible workaround would be to additionally split the returned concatenation of ciphertexts c_b according to the lengths of the ciphertexts in $\mathbf{c}_{1-b}$, and include the resulting set of ciphertexts in $\mathtt{C}$ as well. However we do not know if this definition is satisfiable either. Accordingly it remains an open question whether a meaningful and satisfiable definition of BH-sbbCFA is conceivable or not. More generally, we do not know whether this limitation is due to our inability to formulate such a definition, or because the goal of boundary hiding inherently requires protecting against replay and reordering attacks.

C Auxiliary Security Definitions

C.1 Message Authentication

A *message authentication code* $\mathcal{MA} = (\mathcal{K}, \mathcal{T}, \mathcal{V})$ is a triple of algorithms with an associated message space $\mathcal{M} \subseteq \{0,1\}^*$. The randomised *key-generation* algorithm $\mathcal{K}$ takes no input and returns a secret key K. The deterministic *tagging* algorithm $\mathcal{T}$ takes as input a secret key $K \in \mathcal{K}$, and a message $m \in \mathcal{M}$, and returns a tag $\tau \in \{0,1\}^*$. The deterministic *verification* algorithm $\mathcal{V}$ takes as input the secret key $K \in \mathcal{K}$, a message $m \in \mathcal{M}$, and a candidate tag τ', to return a symbol $v \in \{\mathtt{valid}, \perp\}$ denoting whether τ' is a valid tag for m or not. We require that for any key $K \in \mathcal{K}$ and any $m \in \mathcal{M}$ it hold that:

$$\mathcal{V}_K(m, \mathcal{T}_K(m)) = \mathtt{valid}.$$

A number $\ell_{tag} \geq 1$ is called the *tag length* associated to the scheme if for any key $K \in \mathcal{K}$ and any $m \in \mathcal{M}$

$$|\mathcal{T}_K(m)| = \ell_{tag}.$$

The standard security notion for message authentication schemes is existential unforgeability under chosen message attacks (UF-CMA). This is an adaptation to the symmetric setting of the corresponding notion for signature schemes introduced by Golwasser, Micali, and Rivest [13]. An adversary is allowed to obtain tags for some number of messages of its choice, and wins if it can output a *new* message together with a valid tag. A more stringent security requirement on a MAC is that the tagging algorithm be a pseudorandom function. Both definitions are defined more formally below.

```
$\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{SE}}(\mathcal{A})$
    $K \leftarrow \mathcal{K}$
    $\mathsf{L} \leftarrow \emptyset, \mathsf{win} \leftarrow 0$
    $\mathcal{A}^{\mathsf{Tag}(\cdot),\mathsf{Ver}(\cdot,\cdot)}$
    return win

$\mathsf{Tag}(m)$
    $\tau \leftarrow \mathcal{T}_K(m)$
    $\mathsf{L} \leftarrow \mathsf{L} \cup m$
    return $\tau$

$\mathsf{Ver}(m, \tau)$
    $v \leftarrow \mathcal{V}_K(m, \tau)$
    if $v = \mathtt{valid}$ and $m \notin \mathsf{L}$ then
        $\mathsf{win} \leftarrow 1$
    return $v$
```

Figure 20: Experiment to define UF-CMA security.

Definition C.1. [UF-CMA] Let $\mathcal{MA} = (\mathcal{K}, \mathcal{T}, \mathcal{V})$ be a message authentication code. For an adversary $\mathcal{A}$, define experiment $\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A})$ as shown in Figure 20. A key K is first generated by calling $\mathcal{K}$. The adversary $\mathcal{A}$ is then given access to a tagging oracle $\mathsf{Tag}(\cdot)$ and a verification oracle $\mathsf{Ver}(\cdot,\cdot)$. The adversary wins if it makes a successful verification query for some message which it had not previously queried to the tagging oracle. We define the adversary's advantage as:

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}) = \Pr\left[\, \mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}) = 1 \,\right].$$

The message authentication code $\mathcal{MA}$ is said to be UF-CMA secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A})$ is small.

Definition C.2. [Pseudorandom Functions] Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function family indexed by the set $\mathcal{K}$. Consider an adversary $\mathcal{A}$ with oracle access to some function with domain $\mathcal{X}$ and codomain $\mathcal{Y}$, and which returns a single bit as its output. We define the prf-advantage of adversary $\mathcal{A}$ with respect to the function family F as:

$$\mathbf{Adv}^{\text{prf}}_{F}(\mathcal{A}) = \Pr\left[\, K \leftarrow_{\$} \mathcal{K} : \mathcal{A}^{F_K(\cdot)} = 1 \,\right] - \Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(\mathcal{X}, \mathcal{Y}) : \mathcal{A}^{f(\cdot)} = 1 \,\right] ;$$

where $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ denotes the set of all functions with domain $\mathcal{X}$ and codomain $\mathcal{Y}$. F is said to be a pseudorandom function (PRF), if for every adversary $\mathcal{A}$ with reasonable resources its prf-advantage $\mathbf{Adv}^{\text{prf}}_{F}(\mathcal{A})$ is small.

C.2 Indistinguishability from Random Strings

Definition C.3. [IND\$-CPA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$ and a bit b, define the experiment $\mathbf{Exp}^{\text{ind\$-cpa-b}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 21. The experiment starts by calling

```
$\mathbf{Exp}^{\text{ind\$-cpa-b}}_{\mathcal{SE}}(\mathcal{A})$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, \mathsf{C} \leftarrow ()$
    $b' \leftarrow \mathcal{A}^{\mathsf{Enc\$}(\cdot)}$
    return $b'$

$\mathsf{Enc\$}(m)$
    $(c, \sigma) \leftarrow \mathcal{E}_K(m, \sigma)$
    if $b = 0$ then $c \leftarrow_{\$} \{0,1\}^{|c|}$
    $i \leftarrow i + 1, \mathsf{C}_i \leftarrow c$
    return $c$
```

Figure 21: Experiment to define IND\$-CPA security.

$\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to a special encryption oracle $\mathsf{Enc\$}(\cdot)$. If $b = 1$ the oracle returns the encrypted message, otherwise it returns a uniformly-random bit-string of the same length as the encrypted message.

The adversary's goal is to output a bit b' as its guess of the challenge bit b, and the experiment returns b' as well. The corresponding advantage of an adversary $\mathcal{A}$ is given by:

$$\mathbf{Adv}_{\mathcal{SE}}^{\text{ind\$-cpa}}(\mathcal{A}) = \Pr\left[\,\mathbf{Exp}_{\mathcal{SE}}^{\text{ind\$-cpa-1}}(\mathcal{A}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}_{\mathcal{SE}}^{\text{ind\$-cpa-0}}(\mathcal{A}) = 1\,\right],$$

The scheme $\mathcal{SE}$ is said to be $\mathsf{IND\$\text{-}CPA}$ secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}_{\mathcal{SE}}^{\text{ind\$-cpa}}(\mathcal{A})$ is small.