Simulatable Channels: Extended Security that is Universally Composable and Easier to Prove

Jean Paul Degabriele Marc Fischlin

Cryptoplexity, Technische Universität Darmstadt, Germany
`www.cryptoplexity.de`
`jeanpaul.degabriele@cryptoplexity.de` `marc.fischlin@cryptoplexity.de`

Abstract. Ever since the foundational work of Goldwasser and Micali, simulation has proven to be a powerful and versatile construct for formulating security in various areas of cryptography. However security definitions based on simulation are generally harder to work with than game based definitions, often resulting in more complicated proofs. In this work we challenge this viewpoint by proposing new simulation-based security definitions for secure channels that in many cases lead to simpler proofs of security. We are particularly interested in definitions of secure channels which reflect real-world requirements, such as, protecting against the replay and reordering of ciphertexts, accounting for leakage from the decryption of invalid ciphertexts, and retaining security in the presence of ciphertext fragmentation. Furthermore we show that our proposed notion of channel simulatability implies a secure channel functionality that is universally composable. To the best of our knowledge, we are the first to study universally composable secure channels supporting these extended security goals. We conclude, by showing that the Dropbear implementation of SSH-CTR is channel simulatable in the presence of ciphertext fragmentation, and therefore also realises a universally composable secure channel. This is intended, in part, to highlight the merits of our approach over prior ones in admitting simpler security proofs in comparable settings.

Keywords. Secure Channels · Ciphertext Fragmentation · Universal Composability · SSH · Subtle Authenticated Encryption

1 Introduction

Over the years, several security notions for symmetric encryption have been proposed in the cryptographic literature. In [BDJR97] Bellare *et al.* studied four notions of confidentiality: semantic security, find-then-guess security, left-or-right security, and real-or-random security, and showed them to be all equivalent. Another notion, used in [AR07], demands indistinguishability between encryptions of real messages and encryptions of some fixed message of the same length. This is known to be equivalent to the other four definitions and indeed we will make extensive use of it in this work. Perhaps the most popular notion of confidentiality today is indistinguishability from random bits, often denoted as $\mathsf{IND\$\text{-}CPA}$, which was put forward in [RBBK01, Rog04]. This requires ciphertexts to be indistinguishable from random strings of the same length. In [Rog04] Rogaway gave a number of reasons why he prefers this notion over all others, arguing that it is stronger, easier to prove, yielding more versatile objects, and being conceptually simpler. Indeed these are likely to be the reasons to which this notion owes its popularity.

In our view, however, the aspect that makes IND\$-CPA fundamentally different from all other notions is that it requires the encryption of real messages to be indistinguishable from something computed without any knowledge of the secret key. Thus, at its core is the idea that encryption be *simulatable*, where in this specific case the simulator is required to be of a specific type. The all-in-one notion of authenticated encryption introduced in [RS06], requiring indistinguishability of the encryption from $\$(\cdot)$ and of the decryption from $\perp(\cdot)$, can be similarly viewed as requiring that both processes be simulatable. It is then natural to ask if there is something special about these two specific simulators, or if they can be generalised further.

It turns out that a more general formulation is possible, and this is exactly what we set out to explore in this work. As we shall see, formulating security this way requires some care in order to guarantee the level of security that we expect. In this respect, we identify some necessary restrictions that need to be imposed on the simulators in order to meet their intended goal. We also establish relations between the notions that we propose and also uncover certain interesting connections, for instance, if (and only if) encryption can be simulated by a *stateless* algorithm, then the encryption is key private. In addition, our security notions have the added nice feature that, unlike other security definitions, there are no prohibited queries that the adversary is not allowed to make.

Beyond being of theoretical interest, there is also a more pragmatic reason motivating our study of these security notions. We are primarily interested in symmetric encryption with advanced properties such as protecting against replay and reordering of ciphertexts, maintaining security in the presence of inadvertent leakage from invalid ciphertexts, and supporting ciphertext fragmentation. Such properties are particularly relevant to the security of encryption schemes that are deployed in practice. A number of prior works [BKN02, PW10, BDPS12, BDPS14, ABL$^+$14, HKR15, BPS15, FGMP15, ADHP16] have provided treatments of symmetric encryption with such properties, some of which are rather intricate. We believe that our corresponding security definitions, based on simulation, can help to tame this complexity. For instance, most works treat chosen ciphertext security and ciphertext integrity separately. One reason for this is that the all-in-one notion of authenticated encryption does not lend itself well to these extended settings. In particular, indistinguishability from random strings is too strong a requirement. In practice ciphertexts will be encoded or prepended with additional fields that render them easy to distinguish. In the presence of ciphertext fragmentation [PW10, BDPS12, ADHP16], this is particularly hard to achieve since it implies that ciphertext boundaries should remain hidden. However, because decryption can now process ciphertexts in a bit-by-bit fashion, ciphertext boundaries are implicitly demarcated by the point at which decryption returns an output. Another complication is that the combination of chosen plaintext security and ciphertext integrity, embodied by the all-in-one notion, no longer implies chosen ciphertext security for schemes which may return more than one error message [BDPS14]. Our notion of *channel simulatability with Integrity*, which can be viewed as a generalisation of the all-in-one notion of Rogaway and Shrimpton, overcomes all these limitations. Another reason why our notions are easier to work with is that they bring the security goal closer to the starting point. Our goal in a security proof will now be to transform the scheme into a simulated one, but because the structure that this simulator needs to satisfy is very loose, it will normally require fewer and simpler steps.

Yet another perk of channel simulatability, is that it also guarantees universal composability. More precisely, we show that a scheme being channel simulatable with integrity implies that it realises a universally composable secure channel. In particular, it is universally composable even when leakage from invalid ciphertexts and ciphertext fragmentation are taken into account. Moreover, channel simulatability is conceptually much simpler and easier to use than the universal composability framework.

We conclude by presenting a proof that the Dropbear SSH-CTR implementation satisfies channel simulatability with integrity. In a recent measurement study [ADHP16] it was found that Dropbear is the most ubiquitous SSH implementation on the Internet, with counter mode being the preferred choice of ciphersuite – hence our choice to analyse this scheme. The security of SSH-CTR, in the case of OpenSSH, was analysed by Paterson and Watson in [PW10]. While the difference between the two implementations is not major and their treatment did take ciphertext fragmentation and multiple errors into account, their security model had some limitations which were pointed out and addressed in [BDPS12, ADHP16]. Furthermore, our treatment guarantees universal composability, which is not known to be implied by any of the prior works. However, we mostly intend this result to serve as testament to the simplicity of our approach and invite the reader to contrast our proof with that in [PW10].

2 Preliminaries

We start by surveying some prior related works, which we will later build upon.

2.0.1 Leakage From Invalid Ciphertexts

In most padding-oracle attacks, such as [CHVV03, DP10, AP13], information is leaked to the adversary during the decryption of invalid ciphertexts rather than valid ones. Consequently such attacks are not captured by the usual security models where invalid ciphertexts invariably generate the same error symbol. This motivated Boldyreva *et al.* to revisit the theory of authenticated encryption in the case where distinguishable error symbols may be returned [BDPS14]. In [ABL$^+$14] Andreeva *et al.* set out to model the case where the decrypted plaintext, or part thereof, becomes available to the adversary – known as Release of Unverified Plaintext (RUP) security. This work employs a syntax where decryption is split into two algorithms, decryption and verification. Combined with the correctness requirement, this has the undesirable consequence that their security model does not capture padding-oracle attacks, since the padding cannot form part of the released plaintext. Yet in [ABL$^+$14] RUP security was in part motivated by the need to protect against such attacks. A related notion, called Robust Authenticated Encryption (RAE), was put forward in [HKR15] in which the adversary also gets access to a plaintext string even if the ciphertext was deemed invalid. RAE is formulated rather differently however, here a scheme is required to be indistinguishable from a randomly-sampled injection with variable expansion augmented with a leakage simulator. This renders RAE a relatively strict security notion, attainable only by a limited set of schemes that generally require two pass encryption and decryption. The above security notions were unified in [BPS15], for the case of nonce-based encryption, under the name Subtle Authenticated Encryption. Here a nonce-based scheme is augmented with a leakage function, to model the information leaked from the decryption of invalid ciphertexts, due to the scheme's implementation. The usual nonce-based security notions are then augmented by additionally providing the adversary with oracle access to the leakage function. We adopt a syntax similar to Subtle AE, adapted to the secure channel setting. Consequently our security notions do capture leakage from invalid ciphertexts.

2.0.2 Ciphertext Fragmentation

Secure channels realised over TCP/IP need to be able to decrypt ciphertexts that may be fragmented in an arbitrary way. The mechanisms needed to support ciphertext fragmentation have been exploited to break confidentiality in the secure channel realisations of SSH [APW09] and IPsec [DP10] which employ CBC encryption. These attacks exposed a limitation of our security models, notably the affected secure channel realisation in SSH was proven secure in [BKN02] in a model which did not account for ciphertext fragmentation. To amend this Paterson and Watson [PW10] proposed a model which accounted for ciphertext fragmentation and used it to show that when SSH is instantiated with counter mode encryption it is secure in this extended security model. The proposed security definition, however, was closely tied to the SSH design and suffered from a number of other issues which limited its applicability and generality. These issues were addressed in [BDPS12] which studied ciphertext fragmentation more generally and introduced the related security notions of boundary hiding and resilience to denial of service. In [FGMP15] Fischlin *et al.* consider an extended setting where in addition to supporting ciphertext fragmentation, encryption takes as input a stream of data (rather than atomic messages) which it may fragment arbitrarily and encrypt separately. Recently in [ADHP16] Albrecht *et al.* did a measurement study of SSH deployment and then used the framework of [BDPS12] to analyse the security of three newly introduced ciphersuites in OpenSSH. In this work we propose simulation-based security definitions supporting ciphertext fragmentation, following the approach used in [BDPS12, ADHP16].

2.1 Notation

Unless otherwise stated, an algorithm may be randomised. An adversary is an algorithm. For any adversary $\mathcal{A}$ and algorithms $\mathcal{X}, \mathcal{Y}, \ldots$ we use $\mathcal{A}^{\mathcal{X}(\cdot),\mathcal{Y}(\cdot),\cdots} \Rightarrow z$ to denote the process of running $\mathcal{A}$ with fresh coins and oracle access to algorithms $\mathcal{X}, \mathcal{Y}, \ldots$ and returning an output z. By convention the running time of an adversary refers to the sum of its actual running time and the size of its description. We generically refer to the resources of an adversary as any subset of the following quantities: its running time, the number of queries that it makes to its oracles, and the total length (in bits) of its oracle queries. If $\mathcal{S}$ is a set then $|\mathcal{S}|$ denotes its size, and $y \twoheadleftarrow \mathcal{S}$ denotes the process of selecting an element from $\mathcal{S}$ uniformly at random and assigning it to y.

We use % to denote the integer modulo operation. For a bit b and a positive integer n, we denote by b^n the string composed of b repeated n times. With $\{0,1\}^n$ we denote the set of all binary strings of length n, and $\{0,1\}^*$ denotes the set of all binary strings of finite length. The empty string is represented by ε. For any two strings u and v, $|u|$ and $|u|_B$ denote the length of u in bits and bytes, respectively, $u \| v$ denotes their concatenation, $u \oplus v$ denotes their bitwise XOR, $u \preceq v$ denotes the prefix predicate which assumes the value true if and only if there exists $w \in \{0,1\}^*$ such that $v = u \parallel w$. We use $u[i,j]$ to denote the substring of u from bit i to bit j inclusive, where the indexes start at 1 and $*$ points to the end of the string. Similarly, $u[i,j]_B$ denotes the substring from byte i to byte j. If i is a non-negative integer, then $\langle i \rangle_\ell$ denotes the unsigned ℓ-bit canonical binary representation of i. Accordingly, $\langle\cdot\rangle^{-1}$ represents the inverse mapping which maps strings of any length to $\mathbb{N}$. We use $\{0,1\}^{**}$ to denote the set of all string sequences.

In every experiment where an adversary interacts with an encryption oracle (real or simulated), we assume that a transcript is maintained of its queries and responses. More specifically, a transcript T is an ordered list of message-ciphertext pairs (m, c), where each entry corresponds to an encryption query. We endow this list with a $\mathsf{next}()$ method which returns its entries, one entry per call, in the same order in which they were created – similarly to a queue. Other times, we will treat T as a set and test whether a specific pair (m, c) is in T. When present in an experiment, the sync flag is initially set to true.

It is often convenient to write distinguishing advantages in a compact form. That is, given an adversary $\mathcal{A}$ which interacts with oracles $\mathcal{X}_1, \mathcal{X}_2$ or with oracles $\mathcal{Z}_1, \mathcal{Z}_2$, we write

$$\underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{X}_1,\mathcal{X}_2\\ \mathcal{Z}_1,\mathcal{Z}_2\end{bmatrix} := \left|\mathrm{Prob}\left[\mathcal{A}^{\mathcal{X}_1,\mathcal{X}_2} \Rightarrow 1\right] - \mathrm{Prob}\left[\mathcal{A}^{\mathcal{Z}_1,\mathcal{Z}_2} \Rightarrow 1\right]\right|.$$

According to this notation we can for example apply the triangle inequality

$$\begin{aligned}&\left|\mathrm{Prob}\left[\mathcal{A}^{\mathcal{X}_1,\mathcal{X}_2} \Rightarrow 1\right] - \mathrm{Prob}\left[\mathcal{A}^{\mathcal{Z}_1,\mathcal{Z}_2} \Rightarrow 1\right]\right|\\ &\quad\leq \left|\mathrm{Prob}\left[\mathcal{A}^{\mathcal{X}_1,\mathcal{X}_2} \Rightarrow 1\right] - \mathrm{Prob}\left[\mathcal{A}^{\mathcal{Y}_1,\mathcal{Y}_2} \Rightarrow 1\right]\right|\\ &\qquad + \left|\mathrm{Prob}\left[\mathcal{A}^{\mathcal{Y}_1,\mathcal{Y}_2} \Rightarrow 1\right] - \mathrm{Prob}\left[\mathcal{A}^{\mathcal{Z}_1,\mathcal{Z}_2} \Rightarrow 1\right]\right|\end{aligned}$$

and write

$$\Delta_{\mathcal{A}} \begin{bmatrix} \mathcal{X}_1, \mathcal{X}_2 \\ \mathcal{Z}_1, \mathcal{Z}_2 \end{bmatrix} \leq \Delta_{\mathcal{A}} \begin{bmatrix} \mathcal{X}_1, \mathcal{X}_2 \\ \mathcal{Y}_1, \mathcal{Y}_2 \end{bmatrix} + \Delta_{\mathcal{A}} \begin{bmatrix} \mathcal{Y}_1, \mathcal{Y}_2 \\ \mathcal{Z}_1, \mathcal{Z}_2 \end{bmatrix}.$$

Similarly, if an adversary $\mathcal{A}'$ simulates oracles $\mathcal{X}_2$ resp. $\mathcal{Z}_2$ to $\mathcal{A}$ through some other oracles $\mathcal{X}_2'$ resp. $\mathcal{Z}_2'$ by modifying the answers, e.g., if $\mathcal{X}_2$ and $\mathcal{Z}_2$ output truncated answers of $\mathcal{X}_2'$ and $\mathcal{Z}_2'$, but otherwise executes $\mathcal{A}$, then we can write

$$\Delta_{\mathcal{A}} \begin{bmatrix} \mathcal{X}_1, \mathcal{X}_2 \\ \mathcal{Z}_1, \mathcal{Z}_2 \end{bmatrix} \leq \Delta_{\mathcal{A}} \begin{bmatrix} \mathcal{X}_1, \mathcal{X}_2' \\ \mathcal{Z}_1, \mathcal{Z}_2' \end{bmatrix}.$$

Note that, strictly speaking, the right hand side considers adversary $\mathcal{A}'$, but since this adversary only adapts the oracle replies we take this already into account by using the other oracles in the notation. Moreover, in all cases, $\mathcal{A}'$ will consume the same resources as $\mathcal{A}$, except for a small overhead in its running time to adapt the oracle queries and responses. Since this overhead is usually minor in comparison to the overall running time, we ignore it.

2.1.1 Syntax.

We consider two types of symmetric encryption, atomic encryption [BDJR97, BKN02] and encryption supporting ciphertext fragmentation [BDPS12, ADHP16]. In both cases we allow invalid ciphertexts to leak information to the adversary, as in *Subtle AE* [BPS15]. However, in contrast to *Subtle AE* our focus is on symmetric channels rather than nonce-based symmetric encryption. We view the latter as a stepping stone to building the former, and we believe that the utility of our security definitions manifests itself when considering symmetric encryption with more complex functionalities than nonce-based encryption.

An *atomic symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of algorithms:

- The randomised key generation algorithm $\mathcal{K}$ returns a secret key K. We will slightly abuse notation and use $\mathcal{K}$ to also identify the key space associated to the key generation algorithm.
- The encryption algorithm $\mathcal{E} : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^*$, may be randomised, stateful or both. It takes as input the secret key $K \in \mathcal{K}$, a plaintext message $m \in \{0,1\}^*$, and returns a ciphertext in $\{0,1\}^*$. For stateful versions it may update its internal state when executed.
- The decryption algorithm $\mathcal{D} : \mathcal{K} \times \{0,1\}^* \to (\{\top, \bot\} \times \{0,1\}^*)$ is deterministic and may be stateful. It takes the secret key K, a ciphertext $c \in \{0,1\}^*$, to return a tuple (v, m) such that $v \in \{\top, \bot\}$ indicates the validity of the corresponding ciphertext and m is a binary string representing a message or some leakage. It may update its state upon execution.

Note that decryption may either return $(\top, m)$, indicating that the ciphertext was valid and decrypts to the message $m \in \{0,1\}^*$, or $(\bot, m)$, indicating that the ciphertext was invalid where $m \in \{0,1\}^*$ may represent an error message, some internal value, or some other form of leakage. The leakage-free setting is modeled by returning $(\bot, \varepsilon)$ in response to an invalid ciphertext.

We further require that an atomic encryption scheme satisfies the following standard correctness condition. We write $c_1, \ldots, c_n \leftarrow \mathcal{E}_K(m_1, \ldots, m_n)$ as shorthand to denote the sequence of encryption operations $c_1 \leftarrow \mathcal{E}_K(m_1), c_2 \leftarrow \mathcal{E}_K(m_2), \ldots, c_n \leftarrow \mathcal{E}_K(m_n)$. Similarly, $(v_1, m_1'), \ldots, (v_n, m_n') \leftarrow \mathcal{D}_K(c_1, \ldots, c_n)$ denotes the analogous sequence of decryption operations.

Definition 2.1 (Atomic Correctness) *For all keys K output by $\mathcal{K}$ and all message sequences $m_1, \ldots, m_n \in \{0,1\}^{**}$, if $c_1, \ldots, c_n \leftarrow \mathcal{E}_K(m_1, \ldots, m_n)$ and $(v_1, m_1'), \ldots, (v_n, m_n') \leftarrow \mathcal{D}_K(c_1, \ldots, c_n)$, then for all $1 \leq i \leq n$ it holds that $v_i = \top$ and $m_i' = m_i$.*

We only require decryption to recover the honestly generated messages when ciphertexts are decrypted in the same order as they were produced. This slightly weaker correctness requirement allows us to cater for schemes with a stateful decryption algorithm.

A *symmetric encryption scheme supporting ciphertext fragmentation* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of algorithms:

- The randomised key generation algorithm $\mathcal{K}$ returns a secret key K. We will slightly abuse notation and use $\mathcal{K}$ to also identify the key space associated to the key generation algorithm.
- The encryption algorithm $\mathcal{E} : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^*$, may be randomised, stateful or both. It takes as input the secret key $K \in \mathcal{K}$, a plaintext message $m \in \{0,1\}^*$, and returns a ciphertext in $\{0,1\}^*$.
- The decryption algorithm $\mathcal{D} : \mathcal{K} \times \{0,1\}^* \to (\{\top, \bot\} \times \{0,1\}^*)^*$ is deterministic and stateful. It takes as input the secret key K and a ciphertext fragment $f \in \{0,1\}^*$, to return a sequence of one or more tuples (v, m) or the empty string. Here $v \in \{\top, \bot\}$ indicates whether the corresponding ciphertext part is valid or not, and m is a binary string representing the recovered message (when $v = \top$) or leakage from an invalid ciphertext (when $v = \bot$).

In contrast to the atomic case, decryption may now return more than one tuple. This is because a ciphertext fragment could be composed of a concatenation of ciphertexts in which case a tuple is returned for each ciphertext. Alternatively, a ciphertext fragment may not contain sufficient information to recover the message or even determine its validity, in which case decryption returns no output. Accordingly, we will generally denote the process of decrypting a ciphertext fragment by $(v_1, m'_1) \dots (v_\ell, m'_\ell) \leftarrow \mathcal{D}_K(f)$, where a single output and no output are indicated by $\ell = 1$ and $\ell = 0$ respectively. Note also that in order to support ciphertext fragmentation decryption must necessarily be stateful.

For schemes supporting ciphertext fragmentation we also require a stronger correctness condition. Namely, decryption should recover the original sequence of messages even when the ciphertexts returned by the encryption algorithm are concatenated together, optionally appended with an arbitrary string, and the result is arbitrarily fragmented into substrings which are individually submitted for decryption in their original order. This is stated formally below, using analogous notation for composite encryption and decryption operations as before.

Definition 2.2 (Correctness Under Ciphertext Fragmentation) *For all keys K output by $\mathcal{K}$, all message sequences $m_1, \dots, m_n \in \{0,1\}^{**}$, and all ciphertext fragment sequences $f_1, \dots, f_k \in \{0,1\}^{**}$, if $c_1, \dots, c_n \leftarrow \mathcal{E}_K(m_1, \dots, m_n)$ and $(v_1, m'_1) \dots (v_\ell, m'_\ell) \leftarrow \mathcal{D}_K(f_1, \dots, f_k)$, where $c_1 \parallel \dots \parallel c_n \preceq f_1 \parallel \dots \parallel f_k$, then it holds that $m'_i = m_i$ and $v_i = \top$ for all $1 \leq i \leq n$.*

2.1.2 A Note on Our Choice of Syntax

Our syntax for schemes supporting ciphertext fragmentation differs from that used in [BDPS12, ADHP16] in three main ways. The most significant difference is that our syntax is more restrictive about how decryption should behave. The syntax in [BDPS12, ADHP16] allows decryption to return a message in separate chunks, similarly to online decryption [HRRV15]. Moreover, what chunk of the message is returned, and when, may vary from scheme to scheme for a given sequence of ciphertext fragments. The only requirement is that the concatenation of the outputs be an encoding of the original sequence of messages. In our case, we ultimately want to relate our security notion to an ideal functionality in the UC framework. Specifying such a functionality forces us to choose a concrete output behaviour for decryption. We opted for a functionality where the message is returned all at once, which is how protocols like TLS and SSH behave in practice. This choice is reflected in our syntax, which allows for slightly simpler security definitions. We encounter a similar issue if we try to extend encryption to take a stream as its input [FGMP15]. We would again be forced to decide on a specific functionality regarding how the plaintext stream is to be fragmented. The most natural and common choice in practice, is to separately encrypt each message fragment as soon as it is input to the encryption algorithm. In turn this would yield a syntax that is equivalent to the one we already have.

The other two differences, however, are merely cosmetic. Instead of decryption returning error symbols from some set $\{\perp_1, \perp_2, \dots\}$, decryption now returns $\perp$ together with a string. Clearly this is without loss of generality, as the former case can be easily be mapped to the latter. Thirdly, due to the differences we just described, the end of message symbol (¶), previously used to delineate message boundaries in the decryption output, becomes redundant in our setting and we therefore drop it.

One notable exception that is not captured by our syntax is the InterMAC scheme, described in [BDPS12], which does exhibit an online decryption behaviour. It should be possible to formulate a different ideal functionality, that reflects InterMAC's behaviour, and replicate our general approach for that setting. However, we do not pursue that direction in this work.

2.2 Security Without Simulation

For atomic encryption schemes we consider two types of security, *plain* and *stateful.* The plain notions of confidentiality and integrity are IND-CCA and INT-CTXT, which correspond to the similarly named notions from Bellare and Namprempre [BN00] extended to the (stronger) *subtle* security setting of [BPS15], where subtleties refer to leakage from different error messages or release of unverified plaintexts. Note that subtle security follows directly from our extended syntax rather than any specific alteration in the security definitions. Stateful notions of confidentiality (IND-sfCCA) and integrity (INT-sfCTXT) were introduced in [BKN02] to additionally protect against the replay and reordering of ciphertexts. Again, through our choice of syntax, we here extend these stateful notions to the subtle setting. We emphasize that our syntax of atomic encryption schemes requires neither encryption nor decryption to be stateful. However the decryption algorithm must be stateful in order for a scheme to satisfy stateful security – hence the name. For schemes supporting ciphertext fragmentation the confidentiality and integrity analogues are IND-sfCFA and INT-sfCFRG from [BDPS12, ADHP16] which we here adapt to our syntax. In all three cases, the weaker IND-CPA notion is the usual one since it is unaffected by subtle security, stateful security, or ciphertext fragmentation.

```
Dec($c'$)
  $(v, m') \leftarrow \mathcal{D}_K(c')$
  if $\exists\, m$ s.t. $(m, c') \in \mathsf{T}$
    $(v, m') \leftarrow (\varepsilon, \varepsilon)$
  return $(v, m')$
```

```
sfDec($c'$)
  $(v, m') \leftarrow \mathcal{D}_K(c')$
  if sync
    $(m, c) \leftarrow \mathsf{T}.\mathsf{next}()$
    if $c' = c$
      $(v, m') \leftarrow (\varepsilon, \varepsilon)$
    else
      sync $\leftarrow$ false
  return $(v, m')$
```

```
cfDec($f$)
  $(v_1, m'_1) \dots (v_\ell, m'_\ell) \leftarrow \mathcal{D}_K(f)$
  $F \leftarrow F \parallel f; j \leftarrow 1$
  while sync $\wedge\, j \leq \ell$
    if $\mathsf{T} = [\,]$
      sync $\leftarrow$ false
    else
      $(m, c) \leftarrow \mathsf{T}.\mathsf{next}()$
      $C \leftarrow C \parallel c$
      if $C \preceq F$
        $j \leftarrow j + 1$
      else
        sync $\leftarrow$ false
  return $(v_j, m'_j) \dots (v_\ell, m'_\ell)$
```

Figure 1: Decryption oracles for defining IND-CCA, IND-sfCCA, IND-sfCFA, INT-CTXT, INT-sfCTXT, and INT-sfCFRG security. T is a live transcript of the adversary's queries to its encryption oracle containing message-ciphertext pairs.

Definition 2.3 (Confidentiality) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an atomic symmetric encryption scheme. Let algorithms* Dec *and* sfDec *be as specified in Figure 1, then for any adversary $\mathcal{A}$ we define the corresponding* IND-CCA *and* IND-sfCCA *advantages as:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{SE}}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{Dec}(\cdot)} \Rightarrow 1 \right] - \Pr\left[\mathcal{A}^{\mathcal{E}_K(0^{|\cdot|}),\mathsf{Dec}(\cdot)} \Rightarrow 1 \right] \right|,$$

and

$$\mathsf{Adv}^{\mathsf{ind\text{-}sfcca}}_{\mathcal{SE}}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{sfDec}(\cdot)} \Rightarrow 1 \right] - \Pr\left[\mathcal{A}^{\mathcal{E}_K(0^{|\cdot|}),\mathsf{sfDec}(\cdot)} \Rightarrow 1 \right] \right|,$$

where in both cases the probabilities are over $K \twoheadleftarrow \mathcal{K}$ and the algorithms' coin tosses. Alternatively, if $\mathcal{SE}$ is a symmetric encryption scheme supporting ciphertext fragmentation, then for any adversary $\mathcal{A}$ the corresponding IND-sfCFA *advantage is given by:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}sfcfa}}_{\mathcal{SE}}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{cfDec}(\cdot)} \Rightarrow 1 \right] - \Pr\left[\mathcal{A}^{\mathcal{E}_K(0^{|\cdot|}),\mathsf{cfDec}(\cdot)} \Rightarrow 1 \right] \right|,$$

where cfDec *is as specified in Figure 1. A scheme $\mathcal{SE}$ is said to be $(\epsilon, \mathcal{R}_{\mathcal{A}})$-*NN *secure, for* NN $\in$ {IND-CCA, IND-sfCCA, IND-sfCFA}*, if for any adversary $\mathcal{A}$ with resources at most $\mathcal{R}_{\mathcal{A}}$, its* NN *advantage is bounded by ϵ.*

In the above definition, $\mathcal{E}_K(0^{|\cdot|})$ is an oracle that on input m returns an encryption of $0^{|m|}$. This formulation of confidentiality is equivalent (up to a small constant factor in the advantages) to the more popular left-or-right and real-or-random formulations.

Definition 2.4 (Ciphertext Integrity) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an atomic symmetric encryption scheme. Let algorithms* Dec *and* sfDec *be as specified in Figure 1 and* FORGE *denote the event that the decryption oracle returns a pair (v, m') where $v = \top$. Then for any adversary $\mathcal{A}$ the corresponding* INT-CTXT *and* INT-sfCTXT *advantages are defined as:*

$$\mathsf{Adv}^{\mathsf{int\text{-}ctxt}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[K \twoheadleftarrow \mathcal{K},\ \mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{Dec}(\cdot)} : \mathsf{FORGE} \right],$$

and

$$\mathsf{Adv}^{\mathsf{int\text{-}sfctxt}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[K \twoheadleftarrow \mathcal{K},\ \mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{sfDec}(\cdot)} : \mathsf{FORGE} \right].$$

Alternatively, if $\mathcal{SE}$ is a symmetric encryption scheme supporting ciphertext fragmentation, let algorithm cfDec *be as specified in Figure 1 and* FORGE *denote the event that the decryption oracle return an output $(v_1, m'_1), \ldots, (v_\ell, m'_\ell)$ where $v_i = \top$ for some $1 \leq i \leq \ell$. Then for any adversary $\mathcal{A}$ the corresponding* INT-sfCFRG *advantage is given by:*

$$\mathsf{Adv}^{\mathsf{int\text{-}sfcfrg}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[K \twoheadleftarrow \mathcal{K},\ \mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{cfDec}(\cdot)} : \mathsf{FORGE} \right],$$

where cfDec *is as specified in Figure 1. A scheme $\mathcal{SE}$ is said to be $(\epsilon, \mathcal{R}_{\mathcal{A}})$-*NN *secure, for* NN $\in$ {INT-CTXT, INT-sfCTXT, INT-sfCFRG}*, if for any adversary $\mathcal{A}$ with resources at most $\mathcal{R}_{\mathcal{A}}$, its* NN *advantage is bounded by ϵ.*

In Section 3 we establish a relation between encryption simulatability and key privacy. Key privacy was considered in [Fis99, AR07] for *stateless* symmetric encryption and then covered more extensively in [BBDP01] for the case of public-key encryption. Our definition of key-privacy roughly follows the definitions used in [Fis99, AR07] but we adapt them to cater for stateful schemes. Roughly speaking, the prior definitions would give the adversary access to two encryption oracles and it would then have to distinguish whether the two oracles use the same key or not. Counter mode encryption would not satisfy this definition since an adversary can easily detect two encryptions under the same key and counter value. However counter mode is meant to be used in a way that never re-uses the same counter value (as even confidentiality would fail in that case) and such a situation should never arise in practice. Accordingly we progress the state of the two encryption oracles simultaneously, by encrypting every message by both instances and return to the adversary only one ciphertext which it is allowed to select via an extra bit b given to the oracle. This is stated more formally below.

Definition 2.5 (Key Privacy) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, atomic or supporting ciphertext fragmentation. Let $\langle \mathcal{O}_0(\cdot), \mathcal{O}_1(\cdot)\rangle(b,m)$ be the exclusive oracle combination described in Figure 2, then for any adversary $\mathcal{A}$ we define its* KP-CPA *advantage as:*

$$\mathsf{Adv}^{\mathsf{kp\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}^{\langle \mathcal{E}_K(\cdot), \mathcal{E}_{\bar{K}}(\cdot)\rangle(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\langle \mathcal{E}_K(\cdot), \mathcal{E}_K(\cdot)\rangle(\cdot,\cdot)} \Rightarrow 1\right]\right|,$$

*where the probabilities are over the choice of $K, \bar{K} \twoheadleftarrow \mathcal{K}$ resp. $K \twoheadleftarrow \mathcal{K}$, and the algorithms' coin tosses. A scheme $\mathcal{SE}$ is said to be $(\epsilon, \mathcal{R}_{\mathcal{A}})$-*KP-CPA *secure, if for any adversary $\mathcal{A}$ with resources at most $\mathcal{R}_{\mathcal{A}}$, its* KP-CPA *advantage is bounded by ϵ.*

```
$\langle \mathcal{O}_0(\cdot), \mathcal{O}_1(\cdot)\rangle(b,m)$
$c_0 \leftarrow \mathcal{O}_0(m)$
$c_1 \leftarrow \mathcal{O}_1(m)$
return $c_b$
```

Figure 2: Exclusive oracle combination used in the KP-CPA security definition.

3 Encryption Simulatability

3.1 Defining Encryption Simulatability

As observed in the introduction, IND\$-CPA security stands out from other definitions of confidentiality in that it employs an encryption oracle ($\$(\cdot)$) that does not make use of the encryption key. In particular, we might ask what is special about it that if encryption is indistinguishable from it, then confidentiality is guaranteed? The absence of the encryption key suggests a notion of encryption simulatability and that perhaps pseudorandomness is not really necessary. Indeed this turns out to be the case, but we are still missing one ingredient. The simulator needs to emulate encryption without any knowledge of the message contents except its length. Otherwise the scheme $m \leftarrow \mathcal{E}_K(m)$ would be trivially simulatable but is clearly insecure. A formal definition of encryption simulatability is given below.

Definition 3.1 (Encryption Simulatability) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, either atomic or supporting ciphertext fragmentation. For an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ we define the corresponding* ES *advantage as:*

$$\mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) = \Pr\left[K \twoheadleftarrow \mathcal{K} : \mathcal{A}^{\mathcal{E}_K(\cdot)} \Rightarrow 1\right] - \Pr\left[K \twoheadleftarrow \mathcal{K} : \mathcal{A}^{\mathcal{S}(|\cdot|)} \Rightarrow 1\right]$$

*The scheme $\mathcal{SE}$ is said to be $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-*ES *secure if there exists a randomised and possibly stateful simulator $\mathcal{S}$, requiring at most $\mathcal{R}_{\mathcal{S}}$ resources per query, such that for any adversary $\mathcal{A}$, requiring at most $\mathcal{R}_{\mathcal{A}}$ resources, its respective advantage $\mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S})$ is bounded by ϵ.*

The presence of a simulator in our definition is perhaps reminiscent of other simulation-based security definitions, such as semantic security and even zero knowledge. Intuitively, encryption simulatability says that interacting with the encryption algorithm should convey no knowledge of the key or the message contents. There are some important differences however. In contrast to semantic security, here the simulator is emulating the encryption algorithm rather than the adversary. The simulator cannot depend on the adversary either, due to the reversed order of quantifiers. Finally, contrary to the case of zero knowledge, here the simulator is not allowed to rewind the adversary.

3.2 Understanding Encryption Simulatability

We motivated ES as a generalisation of $\mathsf{IND\$\text{-}CPA}$, and indeed from the definition it follows straight away that $\mathsf{IND\$\text{-}CPA}$ implies ES for any length-regular scheme. Showing that the reverse implication does not hold, i.e., $\mathsf{ES} \not\Longrightarrow \mathsf{IND\$\text{-}CPA}$ is also straightforward, e.g., if the ciphertext contains redundant 0-bits. Despite the differences we mentioned previously, between semantic security (equivalently $\mathsf{IND\text{-}CPA}$) and ES, the two notions turn out to be equivalent. In essence, for any $\mathsf{IND\text{-}CPA}$ symmetric encryption scheme there exists a *stateful* encryption simulator which samples a fresh key at the beginning and runs the encryption algorithm on that key and a fixed message of the length indicated in its input. This is stated more formally together with the reverse implication in Theorem 3.2.

Theorem 3.2 ($\mathsf{IND\text{-}CPA} \iff \mathsf{ES}$) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme.*

a) Then for any encryption simulator $\mathcal{S}$ it holds that:

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S})\,.$$

b) Furthermore, there exists a stateful encryption simulator $\bar{\mathcal{S}}(\ell)$, which on its first input runs $\bar{K} \twoheadleftarrow \mathcal{K}$ once and responds to every query with $\mathcal{E}_{\bar{K}}(0^{\ell})$, such that:

$$\mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \bar{\mathcal{S}}) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A})\,.$$

Proof. For any adversary $\mathcal{A}$ its $\mathsf{IND\text{-}CPA}$ advantage given by:

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) = \underset{\mathcal{A}}{\Delta} \begin{bmatrix} \mathcal{E}_K(\cdot) \\ \mathcal{E}_K(0^{|\cdot|}) \end{bmatrix}.$$

By the triangle inequality we obtain:

$$\leq \underset{\mathcal{A}}{\Delta} \begin{bmatrix} \mathcal{E}_K(\cdot) \\ \mathcal{S}(|\cdot|) \end{bmatrix} + \underset{\mathcal{A}}{\Delta} \begin{bmatrix} \mathcal{S}(|\cdot|) \\ \mathcal{E}_K(0^{|\cdot|}) \end{bmatrix}.$$

Now the first distinguishing game is exactly the ES game, whereas the second game can be reduced to the ES game. In particular, any query m can be simulated by querying $0^{|m|}$ in the ES game, since $|0^{|m|}| = |m|$. Thus it follows that:

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}).$$

This proves the first part of the theorem, we now prove the other direction. For the given simulator $\bar{\mathcal{S}}$ and any adversary $\mathcal{A}$ we have that:

$$\mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \bar{\mathcal{S}}) = \underset{\mathcal{A}}{\Delta} \begin{bmatrix} \mathcal{E}_K(\cdot) \\ \mathcal{E}_{\bar{K}}(0^{|\cdot|}) \end{bmatrix}.$$

Applying the triangle inequality we obtain:

$$\leq \Delta_{\mathcal{A}} \begin{bmatrix} \mathcal{E}_K(\cdot) \\ \mathcal{E}_K(0^{|\cdot|}) \end{bmatrix} + \Delta_{\mathcal{A}} \begin{bmatrix} \mathcal{E}_K(0^{|\cdot|}) \\ \mathcal{E}_{\bar{K}}(0^{|\cdot|}) \end{bmatrix}.$$

Now note that the first term is exactly the IND-CPA advantage, whereas the second term is zero because the two oracles are distributional identical, i.e. for any sequence of queries they yield identically distributed responses (over the choice of the key and potentially the randomness of the encryption scheme). Thus, the result follows:

$$\mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \bar{\mathcal{S}}) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) + 0\,.$$

□

One could also consider chosen-ciphertext extensions of encryption simulatability (ES-ATK for ATK $\in$ $\{\mathsf{CCA}, \mathsf{sfCCA}, \mathsf{CFA}\}$) by additionally providing the adversary with access to the corresponding decryption oracle from Figure 1. While the first implication extends to these settings, i.e. ES-ATK $\implies$ IND-ATK, the implication in the other direction does not! The reason can be seen in the above proof for the IND-CPA case. In the final step of the proof the second advantage term in the proof is no longer zero when a decryption oracle is available. To see why, consider an IND-CCA scheme where every ciphertext is valid, i.e. decrypts to some string [Des00]. Now modify this scheme such that it uses two keys, one used for encryption and decryption and the other is appended to the ciphertexts during encryption. Decryption now checks whether the correct key is appended to the ciphertext, if so it proceeds to decrypt the rest of the ciphertext and returns an error otherwise. The resulting scheme is still IND-CCA secure but a simulator can only guess the right key with negligible probability. An adversary can distinguish the two cases by modifying the part of the ciphertext which is not the key and observe whether its decryption returns a string or an error message. This separation extends easily to the sfCCA and CFA settings. Thus the equivalence between encryption simulatability and semantic security does not extend to the chosen-ciphertext setting.

Interestingly, if we further require that the simulator be stateless, meaning that it maintains no state and uses independent coins in each call, then encryption simulatability additionally guarantees key privacy. The implication holds for schemes which are either stateless or whose state progression is independent of the coins used, which is usually the case in practice, e.g., if a counter is incremented for each call.

Theorem 3.3 (ES $\wedge$ **Stateless**($\mathcal{S}$) $\implies$ KP-CPA) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme such that $\mathcal{E}$ uses fresh coins on each call, and is either stateless or it progresses its state independently of its coins. Then for a stateless simulator $\mathcal{S}$ using fresh coins on every query and any adversary $\mathcal{A}$, it holds that:*

$$\mathsf{Adv}^{\mathsf{kp\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) \leq 3 \cdot \mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S})\,.$$

Proof. For any adversary $\mathcal{A}$ the KP-CPA advantage is given by:

$$\mathsf{Adv}^{\mathsf{kp\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) = \Delta_{\mathcal{A}} \begin{bmatrix} \langle \mathcal{E}_K(\cdot), \mathcal{E}_{\bar{K}}(\cdot) \rangle(\cdot,\cdot) \\ \langle \mathcal{E}_K(\cdot), \mathcal{E}_K(\cdot) \rangle(\cdot,\cdot) \end{bmatrix}.$$

By the triangle inequality, for any encryption simulator $\mathcal{S}$ we have that:

$$\leq \Delta_{\mathcal{A}} \begin{bmatrix} \langle \mathcal{E}_K(\cdot), \mathcal{E}_{\bar{K}}(\cdot) \rangle(\cdot,\cdot) \\ \langle \mathcal{E}_K(\cdot), \mathcal{S}(|\cdot|) \rangle(\cdot,\cdot) \end{bmatrix} + \Delta_{\mathcal{A}} \begin{bmatrix} \langle \mathcal{E}_K(\cdot), \mathcal{S}(|\cdot|) \rangle(\cdot,\cdot) \\ \langle \mathcal{S}(|\cdot|), \mathcal{S}(|\cdot|) \rangle(\cdot,\cdot) \end{bmatrix}$$
$$+ \Delta_{\mathcal{A}} \begin{bmatrix} \langle \mathcal{S}(|\cdot|), \mathcal{S}(|\cdot|) \rangle(\cdot,\cdot) \\ \langle \mathcal{E}_K(\cdot), \mathcal{E}_K(\cdot) \rangle(\cdot,\cdot) \end{bmatrix}.$$

Each of the above terms can be reduced to the encryption simulatability game. In the first term the reduction (playing against $\mathcal{E}_{\bar{K}}(\cdot)$ or $\mathcal{S}(|\cdot|)$) simulates the first oracle $\mathcal{E}_K(\cdot)$ by sampling an independent encryption key K. In the second term the reduction simulates the second oracle by running its own copy of the simulator. The third reduction is where we require the simulator to be stateless and the encryption algorithm to have a state progression that is independent of its coins. The reduction uses one instance of the simulator to emulate two independent ones, which is only possible if the simulator answers each query independently. Similarly for encryption, if the state progression depends only on the key and the message sequence, then both instances of the left and right oracle will progress through the same sequence of states and can therefore be emulated via a single instance. Thus we obtain:

$$\mathsf{Adv}^{\mathsf{kp\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) \leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_{\bar{K}}(\cdot)\\ \mathcal{S}(|\cdot|)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\\ \mathcal{S}(|\cdot|)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{S}(|\cdot|)\\ \mathcal{E}_K(\cdot)\end{bmatrix} \leq 3\cdot \mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}).$$

□

We emphasise that the above implication necessitates that the simulator be stateless. That is, if the simulator is allowed to be stateful then ES does not imply KP-CPA. In particular, a scheme may leak a fixed portion of its key in its ciphertexts and still be IND-CPA secure. Then by Theorem 3.2 the scheme has a *stateful* encryption simulator, but clearly the scheme is not key private.

3.2.1 A Length-Hiding Variant

Our definition of encryption simulatability could be extended to offer a limited form of length hiding by replacing the length function $|\cdot|$ with a rounding length function $\lceil\cdot\rceil$. This would partition the message space into intervals according to the message length. Then messages of differing lengths but wich fall within the same interval would map to the same input to the simulator. Intuitively, the simulator can now only leak the length interval that the message belongs to but not its precise length. This security notion nicely captures the intended protection against traffic analysis offered by practical schemes which pad messages up to a multiple of the block length or some larger value.

4 Decryption Simulatability

It also makes sense to consider an analogous security notion where decryption is required to be simulatable. Although not stated explicitly, security proofs often involve either simulating part of the decryption oracle or employ a specific type of simulator. Indeed ciphertext integrity can be viewed as requiring the existence of a specific type of decryption simulator—one which returns $\perp$ to every query. Error predictability [FGMP15] and leakage simulation [BPS15] are two other examples where parts of the decryption algorithm is simulated. The notion we propose is a generalisation of these ideas, adapted to the channel setting, where we require the whole decryption algorithm to be simulatable. It also allows us to argue about the chosen ciphertext security of schemes which do not provide ciphertext integrity, such as the schemes proposed in [Des00], where any string constitutes a valid ciphertext but it will decrypt to a random-looking message.

4.1 Defining Decryption Simulatability

When defining decryption simulatability it makes sense to also give the adversary access to the encryption algorithm. Then simulation of decryption requests is only possible if as usual we prohibit the adversary from forwarding the ciphertexts it obtains from the encryption oracle. In this particular case, however, we have an alternative option. We can lift these restrictions from the adversary and instead give the decryption simulator access to a live transcript of the encryption queries. Intuitively, this information is already known to the adversary and should result in an equivalent security notion. However, as it turns out, this intuition is not quite correct. We need to restrict the simulator's access to the transcript in order for security to be preserved.

To see why, consider the classical example where we alter a scheme by appending a redundant bit to the ciphertext during encryption and ignore this bit during decryption. This modification renders the scheme malleable and thereby fails to be $\mathsf{IND\text{-}CCA}$ even if the underlying scheme is. However the resulting scheme does have a decryption simulator if it is given unrestricted access to the encryption transcript. In particular, the decryption simulator could use the transcript to simulate the decryption of ciphertexts which are not in the transcript. More concretely, let us assume that the underlying scheme is $\mathsf{IND\text{-}CPA}$ secure and provides ciphertext integrity. Now, if the encryption of m returned $c \,\|\, 0$ and the adversary queries $c \,\|\, 1$, the simulator can, through the available transcript, detect that this is a mauled ciphertext and return m as its response. Alternatively, if the ciphertext is unrelated to a prior encryption query, the simulator returns $\perp$. Thus, if we were to allow unrestricted access to the transcript, the resulting notion of decryption simulatability would not suffice to reduce $\mathsf{IND\text{-}CCA}$ security to $\mathsf{IND\text{-}CPA}$ security.

To overcome this limitation we will wrap the simulator $\mathcal{S}$ with a *fixed* wrapper algorithm that has access to the transcript and possibly overwrites the outputs of $\mathcal{S}$. Specifically, the wrapper will detect whether a ciphertext corresponds to a prior encryption query and replace the output of $\mathcal{S}$ with the message in the transcript, unnoticeable for the simulator. Equivalently, the resulting algorithm can be viewed as a composite decryption simulator where the wrapper component has access to the transcript but its functionality is fixed and $\mathcal{S}$ has no access to the transcript but its functionality is unrestricted and may depend on the scheme. We consider three different wrappers V, W, and Z, described in Figure 3, each yielding a different notion of decryption simulatability. The first, denoted by DS, is plain decryption simulatability and is intended for atomic schemes. Stateful decryption simulatability (SDS) corresponds to the stateful family of security notions which additionally protect against replay and reordering. Fragmented decryption simulatability (FDS) is intended for schemes supporting ciphertext fragmentation.

Definition 4.1 (Decryption Simulatability) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an atomic symmetric encryption scheme. For an adversary $\mathcal{A}$ and a decryption simulator $\mathcal{S}$ we define the corresponding* DS *and* SDS *advantages as:*

$$\mathsf{Adv}^{\mathsf{ds}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) = \Pr\left[\,\mathcal{A}^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)} \Rightarrow 1\,\right] - \Pr\left[\,\mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{V}[\mathcal{S}](\cdot)} \Rightarrow 1\,\right],$$

and

$$\mathsf{Adv}^{\mathsf{sds}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) = \Pr\left[\,\mathcal{A}^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)} \Rightarrow 1\,\right] - \Pr\left[\,\mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{W}[\mathcal{S}](\cdot)} \Rightarrow 1\,\right].$$

where the probabilities are over $K \twoheadleftarrow \mathcal{K}$ and the algorithms' coin tosses. Alternatively, if $\mathcal{SE}$ is a symmetric encryption scheme supporting ciphertext fragmentation, its corresponding FDS *advantage is given by:*

$$\mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) = \Pr\left[\,\mathcal{A}^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)} \Rightarrow 1\,\right] - \Pr\left[\,\mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{Z}[\mathcal{S}](\cdot)} \Rightarrow 1\,\right].$$

A scheme $\mathcal{SE}$ is said to be $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{NN}$ secure, for $\mathsf{NN} \in \{\mathsf{DS}, \mathsf{SDS}, \mathsf{FDS}\}$, if there exists a randomised and possibly stateful simulator $\mathcal{S}$, requiring at most $\mathcal{R}_{\mathcal{S}}$ resources per query, such that for any adversary $\mathcal{A}$, requiring at most $\mathcal{R}_{\mathcal{A}}$ resources, its respective advantage $\mathsf{Adv}^{\mathsf{nn}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S})$ is bounded by ϵ.

4.2 Decryption Simulatability and Chosen-Ciphertext Security

The next theorem states that, as intended, decryption simulatability suffices to reduce chosen ciphertext security to chosen plaintext security. We here state the theorem for the case of schemes supporting ciphertext fragmentation but analogous results hold for atomic schemes in the plain security setting ($\mathsf{IND\text{-}CPA} \wedge \mathsf{DS} \implies \mathsf{IND\text{-}CCA}$) as well as the stateful security setting ($\mathsf{IND\text{-}CPA} \wedge \mathsf{SDS} \implies \mathsf{IND\text{-}sfCCA}$).

```
$\mathsf{V}[\mathcal{S}](c')$   $\boxed{\overline{\mathsf{V}}[\mathcal{S}](c')}$
  $(v,m') \leftarrow \mathcal{S}(c')$
  if $\exists\ m$ s.t. $(m,c') \in \mathsf{T}$
    $(v,m') \leftarrow (\top,m)$
    $\boxed{(v,m') \leftarrow (\varepsilon,\varepsilon)}$
  return $(v,m')$
```

```
$\mathsf{W}[\mathcal{S}](c')$   $\boxed{\overline{\mathsf{W}}[\mathcal{S}](c')}$
  $(v,m') \leftarrow \mathcal{S}(c')$
  if $\mathsf{sync}$
    $(m,c) \leftarrow \mathsf{T.next}()$
    if $c' = c$
      $(v,m') \leftarrow (\top,m)$
      $\boxed{(v,m') \leftarrow (\varepsilon,\varepsilon)}$
    else
      $\mathsf{sync} \leftarrow \mathsf{false}$
  return $(v,m')$
```

```
$\mathsf{Z}[\mathcal{S}](f)$   $\boxed{\overline{\mathsf{Z}}[\mathcal{S}](f)}$
  $(v_1,m'_1)\ldots(v_\ell,m'_\ell) \leftarrow \mathcal{S}(f)$
  $F \leftarrow F \parallel f; j \leftarrow 1$
  while $\mathsf{sync} \wedge j \leq \ell$
    if $\mathsf{T} = [\,]$
      $\mathsf{sync} \leftarrow \mathsf{false}$
    else
      $(m,c) \leftarrow \mathsf{T.next}()$
      $C \leftarrow C \parallel c$
      if $C \preceq F$
        $(v_j,m'_j) \leftarrow (\top,m)$
        $j \leftarrow j+1$
      else
        $\mathsf{sync} \leftarrow \mathsf{false}$
  return $(v_1,m'_1)\ldots(v_\ell,m'_\ell)$
  $\boxed{\textbf{return } (v_j,m'_j)\ldots(v_\ell,m'_\ell)}$
```

Figure 3: The V and W wrappers for an atomic decryption simulator and the Z wrapper for the decryption simulator supporting ciphertext fragmentation, used to define decryption simulatability and channel simulatability. In all three cases the boxed code is omitted. In the suppressing variants $\overline{\mathsf{V}}$, $\overline{\mathsf{W}}$, and $\overline{\mathsf{Z}}$ the boxed lines of code replace the lines above them. T is a live transcript of the adversary's queries to the encryption oracle and is not accessible to $\mathcal{S}$. Note that $(\varepsilon,\varepsilon)$ represents the empty string.

Theorem 4.2 ($\mathsf{IND\text{-}CPA} \wedge \mathsf{FDS} \implies \mathsf{IND\text{-}sfCFA}$) *Let $\mathcal{SE} = (\mathcal{K},\mathcal{E},\mathcal{D})$ be a symmetric encryption scheme supporting ciphertext fragmentation. Then for any adversary $\mathcal{A}$ and any decryption simulator $\mathcal{S}$ it holds that:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}sfcfa}}_{\mathcal{SE}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) + 2 \cdot \mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S})\,.$$

Proof. Observe that the decryption oracle $\mathsf{cfDec}(\cdot)$ in Figure 1 is identical to $\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)$, where $\overline{\mathsf{Z}}$ is described in Figure 3. Then, for any adversary $\mathcal{A}$ its $\mathsf{IND\text{-}sfCFA}$ advantage is given by:

$$\mathsf{Adv}^{\mathsf{ind\text{-}sfcfa}}_{\mathcal{SE}}(\mathcal{A}) = \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\ ,\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\\ \mathcal{E}_K(0^{|\cdot|}),\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\end{bmatrix}.$$

By the triangle inequality, for any decryption simulator $\mathcal{S}$ it holds that:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\\ \mathcal{E}_K(\cdot),\ \overline{\mathsf{Z}}[\mathcal{S}](\cdot)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\ ,\overline{\mathsf{Z}}[\mathcal{S}](\cdot)\\ \mathcal{E}_K(0^{|\cdot|}),\overline{\mathsf{Z}}[\mathcal{S}](\cdot)\end{bmatrix}$$
$$+ \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(0^{|\cdot|}),\ \overline{\mathsf{Z}}[\mathcal{S}](\cdot)\\ \mathcal{E}_K(0^{|\cdot|}),\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\end{bmatrix}.$$

By means of a reduction on the third term we now replace every encryption query m with $0^{|m|}$. Note how this is only possible because the wrapper is suppressing and would not be possible otherwise. In particular, in one case the transcript stores m whereas in the other it stores $0^{|m|}$. However, in both cases the oracle's behaviour is identical since the suppressing wrapper does not make use of the messages in the transcript. We now have that:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\\ \mathcal{E}_K(\cdot),\ \overline{\mathsf{Z}}[\mathcal{S}](\cdot)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\ ,\overline{\mathsf{Z}}[\mathcal{S}](\cdot)\\ \mathcal{E}_K(0^{|\cdot|}),\overline{\mathsf{Z}}[\mathcal{S}](\cdot)\end{bmatrix}$$
$$+ \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\ \overline{\mathsf{Z}}[\mathcal{S}](\cdot)\\ \mathcal{E}_K(\cdot),\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\end{bmatrix}.$$

We now reduce the first and third terms to the FDS game. We employ a straightforward reduction that applies $\overline{\mathsf{Z}}$ to the decryption oracle, and observe that applying $\overline{\mathsf{Z}}$ after Z is equivalent to applying $\overline{\mathsf{Z}}$ directly. This means we can simulate $\overline{\mathsf{Z}}[\mathcal{D}_K]$ resp. $\overline{\mathsf{Z}}[\mathcal{S}]$ through $\mathsf{Z}[\mathcal{D}_K]$ and $\mathsf{Z}[\mathcal{S}]$, and we can then also take advantage of $\mathsf{Z}[\mathcal{D}_K] = \mathcal{D}_K$. Regarding the second term, it can be reduced to IND-CPA by running a local copy of the decryption simulator and wrapper. This yields:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)\\ \mathcal{E}_K(\cdot),\mathsf{Z}[\mathcal{S}](\cdot)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\\ \mathcal{E}_K(0^{|\cdot|})\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\mathsf{Z}[\mathcal{S}](\cdot)\\ \mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)\end{bmatrix},$$
$$= \mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}) + \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}).$$

□

Note that chosen ciphertext security does not imply decryption simulatability, i.e. IND-CCA $\not\Longrightarrow$ DS. To show this separation we can use again the same counterexample that we used in the discussion following Theorem 3.2. That is, a scheme can leak part of the key in its ciphertext and still be IND-CCA secure. Then decryption can behave differently, by returning a string or an error message, depending on whether a ciphertext contains the right key or not. Now, since a decryption simulator does not know the key, it cannot successfully emulate this behaviour and is therefore not DS secure. However, for the case of encryption simulatability the implication is valid, that is, ES-CCA $\Longrightarrow$ DS. In particular, we can simulate decryption by running the algorithm on an independently sampled key. Thus, if encryption is simulatable to an adversary with oracle access to decryption, it follows that decryption is simulatable to an adversary with oracle access to encryption. Analogous relations hold for stateful security and schemes supporting ciphertext fragmentation. Below we state more formally, with proof, the relation for the fragmentation setting.

Theorem 4.3 (ES-sfCFA $\Longrightarrow$ FDS) *Let $\mathcal{SE} = (\mathcal{K},\mathcal{E},\mathcal{D})$ be a symmetric encryption scheme supporting ciphertext fragmentation. Then there exists a stateful decryption simulator $\mathcal{S}_D(c)$, which on its first input runs $\bar{K} \twoheadleftarrow \mathcal{K}$ and responds to every query using $\mathcal{D}_{\bar{K}}(c)$, such that for any encryption simulator $\mathcal{S}_E$ it holds that:*

$$\mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}_D) \leq 2\cdot \mathsf{Adv}^{\mathsf{es\text{-}sfcfa}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}_E)\,.$$

Proof. For the given simulator $\mathcal{S}_D$, which decrypts under a freshly chosen key $\bar{K}$, and any adversary $\mathcal{A}$ the FDS advantage is given by:

$$\mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}_D) = \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\ \mathcal{D}_K(\cdot)\\ \mathcal{E}_K(\cdot),\mathsf{Z}[\mathcal{S}_D](\cdot)\end{bmatrix} = \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\ \mathcal{D}_K(\cdot)\\ \mathcal{E}_K(\cdot),\mathsf{Z}[\mathcal{D}_{\bar{K}}](\cdot)\end{bmatrix}.$$

By the triangle inequality, for any encryption simulator $\mathcal{S}_E$ it holds that:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\ ,\ \mathcal{D}_K(\cdot)\\ \mathcal{S}_E(|\cdot|),\mathsf{Z}[\mathcal{D}_K](\cdot)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{S}_E(|\cdot|),\mathsf{Z}[\mathcal{D}_K](\cdot)\\ \mathcal{E}_K(\cdot)\ ,\mathsf{Z}[\mathcal{D}_{\bar{K}}](\cdot)\end{bmatrix}.$$

By the correctness of the scheme, we can replace $\mathcal{D}_K(\cdot)$ by $\mathsf{Z}[\mathcal{D}_K](\cdot)$ in the upper row of the first term. With respect to the second term we drop the decryption oracle since it can be simulated locally by sampling an independent key and maintaining a local transcript for simulating the wrapper. We thus have:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\ ,\mathsf{Z}[\mathcal{D}_K](\cdot)\\ \mathcal{S}_E(|\cdot|),\mathsf{Z}[\mathcal{D}_K](\cdot)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{S}_E(|\cdot|)\\ \mathcal{E}_K(\cdot)\end{bmatrix}.$$

The first term can now be reduced to a similar game employing a suppressing wrapper since the suppressed queries can be answered by maintaining a local copy of the transcript. Therefore:

$$= \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\ ,\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\\ \mathcal{S}_E(|\cdot|),\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{S}_E(|\cdot|)\\ \mathcal{E}_K(\cdot)\end{bmatrix},$$

and the result now follows

$$= \quad \mathsf{Adv}^{\mathsf{es\text{-}sfcfa}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}_E) \quad + \quad \mathsf{Adv}^{\mathsf{es}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}_E).$$

□

4.3 Decryption Simulatability and Ciphertext Integrity

Informally, decryption simulatability says that access to the decryption algorithm is of no use to an adversary, thereby allowing us to reduce chosen ciphertext security to chosen plaintext security. However, by itself, this does not guarantee ciphertext integrity. Luckily, we only need to impose a minor additional requirement on the simulator for it to cover ciphertext integrity. Essentially, the requirement is that the simulator always returns an error for mauled ciphertexts. It then follows that the real decryption algorithm can only deviate from this behaviour with negligible probability. In our definition we conveniently make use of the suppressing variants of the wrapper algorithms, from Figure 3, in order to filter out any ciphertexts that were obtained from the encryption oracle.

Definition 4.4 (Decryption Simulatability with Integrity) *Let $\mathcal{SE} = (\mathcal{K},\mathcal{E},\mathcal{D})$ be an atomic symmetric encryption scheme. Then $\mathcal{SE}$ is said to be $(\epsilon,\mathcal{R}_{\mathcal{S}},\mathcal{R}_{\mathcal{A}})$-DS-I or $(\epsilon,\mathcal{R}_{\mathcal{S}},\mathcal{R}_{\mathcal{A}})$-SDS-I secure, if it is respectively $(\epsilon,\mathcal{R}_{\mathcal{S}},\mathcal{R}_{\mathcal{A}})$-DS or $(\epsilon,\mathcal{R}_{\mathcal{S}},\mathcal{R}_{\mathcal{A}})$-SDS secure, and, in addition, the corresponding simulator $\mathcal{S}$ augmented with $\overline{\mathsf{V}}$ or $\overline{\mathsf{W}}$ respectively never (with probability zero) outputs a pair (v, m') where $v = \top$.*

Similarly, if $\mathcal{SE}$ is a symmetric encryption scheme supporting ciphertext fragmentation it is said to be $(\epsilon,\mathcal{R}_{\mathcal{S}},\mathcal{R}_{\mathcal{A}})$-FDS-I secure if it is $(\epsilon,\mathcal{R}_{\mathcal{S}},\mathcal{R}_{\mathcal{A}})$-FDS secure and its corresponding simulator $\mathcal{S}$ is such that $\overline{\mathsf{Z}}[\mathcal{S}]$ never (with probability zero) returns an output $(v_1, m'_1),\ldots,(v_\ell, m'_\ell)$ where $v_i = \top$ for some $1 \leq i \leq \ell$.

Informally, the above says that the simulator will never return a valid output for a ciphertext that is not in the transcript (DS-I) or once the queries become out of sync (SDS-I and FDS-I). Note that such a property can be verified simply by inspecting the code of the simulator. Thus no additional steps may be required to prove ciphertext integrity if the decryption simulator already satisfies this condition.

The following theorem says that decryption simulatability with integrity implies the usual notions of ciphertext integrity. We prove this only for schemes supporting ciphertext fragmentation, but analogous theorems and proofs hold for the atomic setting, i.e. DS-I $\Longrightarrow$ INT-CTXT and SDS-I $\Longrightarrow$ INT-sfCTXT.

Theorem 4.5 (FDS-I $\Longrightarrow$ INT-sfCFRG) *Let $\mathcal{SE} = (\mathcal{K},\mathcal{E},\mathcal{D})$ be a symmetric encryption scheme supporting ciphertext fragmentation and let $\mathcal{S}$ be a decryption simulator such that it is $(\epsilon,\mathcal{R}_{\mathcal{S}},\mathcal{R}_{\mathcal{A}})$-FDS-I secure. Then $\mathcal{SE}$ is $(\epsilon,\mathcal{R}_{\mathcal{A}})$-INT-sfCFRG secure.*

Proof. Note that $\mathsf{cfDec}(\cdot)$ is identical to $\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)$. Hence for any simulator $\mathcal{S}$ and any adversary $\mathcal{A}$ with at most $\mathcal{R}_\mathcal{A}$ resources, we have that:

$$\underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\mathsf{cfDec}(\cdot)\\ \mathcal{E}_K(\cdot),\ \overline{\mathsf{Z}}[\mathcal{S}](\cdot)\end{bmatrix} = \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\\ \mathcal{E}_K(\cdot),\ \overline{\mathsf{Z}}[\mathcal{S}](\cdot)\end{bmatrix}.$$

Then, by a straightforward reduction that applies $\overline{\mathsf{Z}}$ to the decryption oracle and observing that $\overline{\mathsf{Z}}[\mathsf{Z}[\mathcal{S}]](\cdot)$ is identical to $\overline{\mathsf{Z}}[\mathcal{S}](\cdot)$, it follows that:

$$\begin{aligned}&\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)\\ \mathcal{E}_K(\cdot),\mathsf{Z}[\mathcal{S}](\cdot)\end{bmatrix},\\ &= \quad \mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}).\end{aligned}$$

From the above relation it then follows that:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{int\text{-}sfcfrg}}_{\mathcal{SE}}(\mathcal{A}) &= \Pr\left[\,K \twoheadleftarrow \mathcal{K},\ \mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{cfDec}(\cdot)} : \mathsf{FORGE}\,\right],\\ &\leq \Pr\left[\,\mathcal{A}^{\mathcal{E}_K(\cdot),\overline{\mathsf{Z}}[\mathcal{S}](\cdot)} : \mathsf{FORGE}\,\right] + \mathsf{Adv}^{\mathsf{fds}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}).\end{aligned}$$

Now since $\mathcal{SE}$ is $(\epsilon, \mathcal{R}_\mathcal{S}, \mathcal{R}_\mathcal{A})$-$\mathsf{FDS\text{-}I}$ secure, there exists a simulator such that the first term is zero and the second term is bounded by ϵ, thus:

$$\leq \epsilon.$$

4.3.1 Comparing DS to Prior Notions

We are not the first to consider notions requiring the decryption algorithm to be simulatable. Two notable cases are the works of Andreeva *et al.* [ABL$^+$14] and that of Hoang, Krovetz, and Rogaway [HKR15]. Below is a comparison of our notion with these

Inspired by plaintext awareness the authors of [ABL$^+$14] propose two security notions called PA1 and PA2, which involve an *extractor* algorithm that essentially acts as a decryption simulator. Their first notion, PA1, roughly corresponds to a notion of decryption simulatability where the simulator has unrestricted access to the transcript. As we described in Section 4.1, such a formulation would not suffice to guarantee chosen-ciphertext security and results in a weaker notion. Accordingly, the authors put forward PA2 where the extractor no longer has access to the transcript and the adversary is prohibited from querying ciphertext to the extractor that it obtains from its encryption oracle. We note, however, that a our notions and relations are not directly comparable to those in [ABL$^+$14] since their work assumes a different syntax. Apart from being nonce-based and requiring encryption to be deterministic, their syntax splits decryption into separate decryption and verification algorithms. This choice of syntax has important consequences, where for instance, their resulting IND-CCA notion is weaker than the traditional one, see [BPS15].

A decryption simulator also appears in the definition of Robust Authenticated Encryption (RAE) from [HKR15]. RAE security requires that a (nonce-based) encryption scheme be indistinguishable from an idealised scheme where encryption is a randomly-sampled injection, and decryption can be viewed as answering its queries either by looking up the transcript or via a simulator. That is, the idealised decryption oracle in RAE essentially behaves as our combination of a decryption simulator and wrapper algorithm. Note that in RAE the decryption simulator appears in conjunction with an ideal encryption oracle, whereas in DS it appears in conjunction with the real encryption algorithm. As such, RAE is perhaps more akin to $\mathsf{ES} \wedge \mathsf{DS}$ (discussed in Section 5.1). Indeed, RAE security could be viewed as a special case of $\mathsf{ES} \wedge \mathsf{DS}$ (translated to the nonce-based setting), where the encryption simulator is further restricted to be a pseudorandom injection.

5 Channel Simulatability

We can now go a step further and require that both encryption and decryption be simulatable.

5.1 Defining Channel Simulatability

A natural formulation is to require that there exist an encryption simulator $\mathcal{S}_E$ and a decryption simulator $\mathcal{S}_D$ such that no adversary can distinguish between unrestricted oracle access to $\mathcal{E}_K(\cdot)$ and $\mathcal{D}_K(\cdot)$ or $\mathcal{S}_E(|\cdot|)$ and $\mathsf{V}[\mathcal{S}_D](\cdot)$. Such a notion turns out to be equivalent to $\mathsf{ES}\wedge\mathsf{DS}$, i.e. the requirement that a scheme satisfy both simulatability notions ES and DS. This notion can be viewed as a stronger analogue of $\mathsf{IND\text{-}CCA}$ security. Indeed, because decryption simulatability reduces $\mathsf{IND\text{-}CCA}$ security to $\mathsf{IND\text{-}CPA}$ security and encryption simulatability implies $\mathsf{IND\text{-}CPA}$, it follows that $\mathsf{ES} \wedge \mathsf{DS} \implies \mathsf{IND\text{-}CCA}$. Similarly $\mathsf{ES} \wedge \mathsf{DS\text{-}I}$, where decryption simulatability also ensures integrity, can be viewed as an analogue and a generalisation of the combined authenticated encryption security notion from [RS06]. Clearly, all of the above also holds for stateful security ($\mathsf{ES} \wedge \mathsf{SDS\text{-}I}$) and for schemes supporting ciphertext fragmentation ($\mathsf{ES} \wedge \mathsf{FDS\text{-}I}$).

We believe these notions are appealing for a number of reasons. On an intuitive level, these notions say that an adversary's computational abilities are not any better when it is given oracle access to the channel, since it can be simulated. That is, the ability to choose the messages that get encrypted, replay, reorder and fragment ciphertexts arbitrarily, and observe the output of the decryption algorithm (possibly augmented with additional leakage such as error messages and the release of unverified plaintext) are of no help to the adversary. Moreover, there are no prohibited or suppressed queries, as is the case with all CCA and authenticated encryption type of definitions. Being single-game definitions, they are also easier to prove than their two-game counterparts used in [BKN02, PW10, BDPS12, FGMP15, ADHP16]. Further backing to the claim that these notions are easier to prove can be found in Section 7. Finally, as we will show later on, any scheme that meets these notions realises a universally composable secure channel. Thus our notions guarantee composability under extended security requirements, such as the presence of leakage from invalid ciphertexts [BDPS14, ABL$^+$14, HKR15, BPS15], protection against replay and reordering [BKN02], and security in the presence of ciphertext fragmentation [PW10, BDPS12, FGMP15, ADHP16].

However the above formulation, requiring separate simulators, has some limitations. For instance the schemes used in SSH, which include an encrypted length field as part of their ciphertext – see Section 7 or [PW10, ADHP16], cannot meet this notion. In particular, because a ciphertext may be delivered as multiple fragments, the length field is used by the decryption algorithm to determine the total length of the ciphertext and accordingly at which point to verify the MAC tag. As such the decryption simulator needs to be able to predict, both for in-sync and out-of-sync ciphertexts, after how many bytes it should return an output. Note that the contents of length field are known to the adversary and any inconsistency between the real scheme and the simulated one would allow it to distinguish the two. At the same time, the encryption simulator cannot leak this information anywhere in the ciphertext, except through its size, as otherwise it would either not constitute a good simulator, or the encryption used to protect the length field in the real scheme is insecure. Consequently, for the schemes used in SSH there can exist no pair of simulators that satisfy the security definition outlined above.

In the case of $\mathsf{SSH\text{-}CTR}$ this issue can be overcome by allowing the simulators to share a random tape that they can then use to one-time-pad the length field. In general, the more freedom we give the simulators to share resources and communicate the easier it becomes to satisfy such a security notion. We therefore lift all such restrictions by replacing the two simulators with a single simulator having separate interfaces for encryption and decryption, $\mathcal{S}(\mathsf{e}, \cdot)$ and $\mathcal{S}(\mathsf{d}, \cdot)$. The resulting notion, which we call channel simulatability (CS) is stated more formally in Defintion 5.1 and in Defintion 5.3. Note that $\mathsf{ES} \wedge \mathsf{DS} \implies \mathsf{CS}$ since two separate simulators can easily be combined into one, but the converse is not true. While it is easy to see that channel simulatability retains the appealing properties that we mentioned earlier, the SSH example we just described separates it from $\mathsf{ES} \wedge \mathsf{DS}$. We must therefore make sure that channel simulatability still offers an adequate level of security. We assert this in Theorem 5.2 and Theorem 5.3, where we prove that it guarantees chosen ciphertext security and integrity. The results are stated for schemes supporting ciphertext fragmentation but analogous results hold in the atomic setting for plain and stateful security. In Section 6 we show that channel simulatability implies UC-realising the secure channel ideal functionality. By transitivity, it follows that $\mathsf{ES} \wedge \mathsf{DS}$ also guarantees universal composability.

Definition 5.1 (Channel Simulatability) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For any adversary $\mathcal{A}$ and a channel simulator $\mathcal{S}$ we define the corresponding* CS *and* SCS *advantages as:*

$$\mathsf{Adv}^{\mathsf{cs}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) = \Pr\left[\, \mathcal{A}^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \mathcal{A}^{\mathcal{S}(\mathsf{e}, |\cdot|), \mathsf{V}[\mathcal{S}](\mathsf{d}, \cdot)} \Rightarrow 1 \,\right],$$

and,

$$\mathsf{Adv}^{\mathsf{scs}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) = \Pr\left[\, \mathcal{A}^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \mathcal{A}^{\mathcal{S}(\mathsf{e}, |\cdot|), \mathsf{W}[\mathcal{S}](\mathsf{d}, \cdot)} \Rightarrow 1 \,\right],$$

where the probabilities are over $K \twoheadleftarrow \mathcal{K}$ and the algorithms' coin tosses. Alternatively, if $\mathcal{SE}$ is a symmetric encryption scheme supporting ciphertext fragmentation, its corresponding FCS *advantage is given by:*

$$\mathsf{Adv}^{\mathsf{fcs}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) = \Pr\left[\, \mathcal{A}^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \mathcal{A}^{\mathcal{S}(\mathsf{e}, |\cdot|), \mathsf{Z}[\mathcal{S}](\mathsf{d}, \cdot)} \Rightarrow 1 \,\right].$$

A scheme $\mathcal{SE}$ is said to be $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{NN}$ secure, for $\mathsf{NN} \in \{\mathsf{CS}, \mathsf{SCS}, \mathsf{FCS}\}$, if there exists a randomised and possibly stateful simulator $\mathcal{S}$ such that every query of the form $\mathcal{S}(\mathsf{e}, \cdot)$ or $\mathcal{S}(\mathsf{d}, \cdot)$ requires at most $\mathcal{R}_{\mathcal{S}}$ resources, and for any adversary $\mathcal{A}$, requiring at most $\mathcal{R}_{\mathcal{A}}$ resources, its respective advantage $\mathsf{Adv}^{\mathsf{nn}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S})$ is bounded by ϵ.

Theorem 5.2 ($\mathsf{FCS} \implies \mathsf{IND\text{-}sfCFA}$) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme supporting ciphertext fragmentation. Then for any adversary $\mathcal{A}$ and any channel simulator $\mathcal{S}$ it holds that:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}sfcfa}}_{\mathcal{SE}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{fcs}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}).$$

Proof. Observing that $\mathsf{cfDec}(\cdot)$ is identical to $\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)$, it follows that for any adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{ind\text{-}sfcfa}}_{\mathcal{SE}}(\mathcal{A}) = \underset{\mathcal{A}}{\Delta}\begin{bmatrix} \mathcal{E}_K(\cdot) \;, \overline{\mathsf{Z}}[\mathcal{D}_K](\cdot) \\ \mathcal{E}_K(0^{|\cdot|}), \overline{\mathsf{Z}}[\mathcal{D}_K](\cdot) \end{bmatrix}.$$

By the triangle inequality, for any channel simulator $\mathcal{S}$ it follows that:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix} \mathcal{E}_K(\cdot) \;, \overline{\mathsf{Z}}[\mathcal{D}_K](\cdot) \\ \mathcal{S}(\mathsf{e}, |\cdot|), \overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d}, \cdot) \end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix} \mathcal{S}(\mathsf{e}, |\cdot|), \overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d}, \cdot) \\ \mathcal{E}_K(0^{|\cdot|}), \overline{\mathsf{Z}}[\mathcal{D}_K](\cdot) \end{bmatrix}.$$

In the second term, since the wrapper is suppressing, we can replace every encryption query m with $0^{|m|}$, reducing it to:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix} \mathcal{E}_K(\cdot) \;, \overline{\mathsf{Z}}[\mathcal{D}_K](\cdot) \\ \mathcal{S}(\mathsf{e}, |\cdot|), \overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d}, \cdot) \end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix} \mathcal{S}(\mathsf{e}, |\cdot|), \overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d}, \cdot) \\ \mathcal{E}_K(\cdot) \;, \overline{\mathsf{Z}}[\mathcal{D}_K](\cdot) \end{bmatrix}.$$

Through a straightforward reduction that applies $\overline{\mathsf{Z}}$ to the decryption oracle and observing that applying $\overline{\mathsf{Z}}$ after Z is equivalent to applying $\overline{\mathsf{Z}}$ directly, we obtain:

$$\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix} \mathcal{E}_K(\cdot) \;, \mathsf{Z}[\mathcal{D}_K](\cdot) \\ \mathcal{S}(\mathsf{e}, |\cdot|), \mathsf{Z}[\mathcal{S}](\mathsf{d}, \cdot) \end{bmatrix} + \underset{\mathcal{A}}{\Delta}\begin{bmatrix} \mathcal{S}(\mathsf{e}, |\cdot|), \mathsf{Z}[\mathcal{S}](\mathsf{d}, \cdot) \\ \mathcal{E}_K(\cdot) \;, \mathsf{Z}[\mathcal{D}_K](\cdot) \end{bmatrix},$$

and the result follows

$$= \quad \mathsf{Adv}^{\mathsf{fcs}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}) \quad + \quad \mathsf{Adv}^{\mathsf{fcs}}_{\mathcal{SE}}(\mathcal{A}, \mathcal{S}).$$

□

5.2 Channel Simulatability with Integrity

Just like decryption simulatability, channel simulatability can easily be extended to guarantee ciphertext integrity by additionally requiring an easily verifiable property from the channel simulator. Informally, we require that, by design, the simulator never return a valid output for a ciphertext that is not in the transcript ($\mathsf{CS\text{-}I}$) or once the queries become out of sync ($\mathsf{SCS\text{-}I}$ and $\mathsf{FCS\text{-}I}$).

Definition 5.3 (Channel Simulatability with Integrity) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an atomic symmetric encryption scheme. Then $\mathcal{SE}$ is said to be $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{CS\text{-}I}$ or $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{SCS\text{-}I}$ secure, if it is respectively $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{CS}$ or $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{SCS}$ secure, and, in addition, the corresponding channel simulator $\mathcal{S}$ is such that $\overline{\mathsf{V}}[\mathcal{S}](\mathsf{d}, \cdot)$,or respectively $\overline{\mathsf{W}}[\mathcal{S}](\mathsf{d}, \cdot)$, never (with probability zero) outputs a pair (v, m') where $v = \top$.*

Similarly, if $\mathcal{SE}$ is a symmetric encryption scheme supporting ciphertext fragmentation it is said to be $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{FCS\text{-}I}$ secure if it is $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{FCS}$ secure and its corresponding simulator $\mathcal{S}$ is such that $\overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d}, \cdot)$ never (with probability zero) returns an output $(v_1, m'_1), \ldots, (v_\ell, m'_\ell)$ where $v_i = \top$ for some $1 \leq i \leq \ell$.

The theorem below states that channel simulatability with integrity implies the respective notion of ciphertext integrity. The theorem is stated for the case of ciphertext fragmentation, but analogous results hold for the atomic schemes. Its proof is similar to that of Theorem 4.5 with some minor adaptations, but for completeness we include it in Appendix A.

Theorem 5.4 ($\mathsf{FCS\text{-}I} \implies \mathsf{INT\text{-}sfCFRG}$) *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme supporting ciphertext fragmentation and let $\mathcal{S}$ be a channel simulator such that it is $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{FCS\text{-}I}$ secure. Then $\mathcal{SE}$ is $(\epsilon, \mathcal{R}_{\mathcal{A}})$-$\mathsf{INT\text{-}sfCFRG}$ secure.*

6 Simulatable Channels and Universal Composability

In this section we show that any scheme satisfying channel simulatability with integrity realises a universally composable channel.

6.1 UC Framework

The universal composition framework [Can01] is a simulation-based security notion for a protocol π implementing some ideal functionality $\mathcal{F}$. The approach requires that for any adversary $\mathcal{A}_{\mathrm{UC}}$ attacking a real protocol π between parties $P_1, P_2, \ldots$ there exists an ideal-model adversary $\mathcal{S}_{\mathrm{UC}}$ (or, simulator) interacting in a world where all parties are connected to the ideal functionality $\mathcal{F}$. The only task of the parties in this ideal world is to forward their inputs to $\mathcal{F}$ and output the responses of $\mathcal{F}$. The communication with the ideal functionality is not visible to other parties and cannot be tampered with.

We give here only an informal introduction to the model and refer to [Can01] for the details. The UC model is different from other simulation-based notions in that it uses an interactive distinguisher to decide in which of the two worlds the execution takes place. This interactive distinguisher is called the environment $\mathcal{E}_{\mathrm{UC}}$, since it represents other potentially ongoing protocols and thereby ensures composability. The environment determines the input of the parties, learns their outputs, and can interact with the (real or ideal) adversary. To distinguish inputs for different sessions, the UC model assumes that globally unique and publicly known session identifiers *sid* are assigned to each protocol execution.

Let $\mathsf{REAL}_{\mathcal{A}_{\mathrm{UC}}, \mathcal{E}_{\mathrm{UC}}, \pi}(n)$ be the random variable denoting the environment's output in a real-world execution, where $\mathcal{A}_{\mathrm{UC}}$ interacts with the protocol π for security parameter n, and $\mathsf{IDEAL}_{\mathcal{S}_{\mathrm{UC}}, \mathcal{E}_{\mathrm{UC}}, \mathcal{F}}(n)$ be the corresponding random variable when interacting with $\mathcal{S}_{\mathrm{UC}}$ in the ideal world. We say that a protocol π *securely realises* $\mathcal{F}$ if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}_{\mathrm{UC}}$ there exists a PPT simulator $\mathcal{S}_{\mathrm{UC}}$ such that for any PPT environment $\mathcal{E}_{\mathrm{UC}}$ the random variables $\mathsf{REAL}_{\mathcal{A}_{\mathrm{UC}},\mathcal{E}_{\mathrm{UC}},\pi}$ and $\mathsf{IDEAL}_{\mathcal{S}_{\mathrm{UC}},\mathcal{E}_{\mathrm{UC}},\mathcal{F}}$ are computationally indistinguishable. For concrete security one would measure the difference in the output distributions exactly. By viewing a potential distinguisher of the environment's output as part of the environment itself, we can equivalently assume that the environment only outputs a bit to indicate which world it is in.

In principle, the adversary may decide to corrupt parties, either at the outset of the execution (static corruptions) or while the execution is running (adaptive corruptions). This choice affects even the design of the ideal functionality. However, for channels based on symmetric keys, corrupting one of the participating parties discloses the secret key and the channel can no longer provide any security guarantees. Hence, we assume that the adversary neither corrupts the sender nor the receiver in our scenario.

Some technicalities of the UC framework are relevant to our setting. The first is that it suffices to consider so-called dummy adversaries $\tilde{\mathcal{A}}_{\mathrm{UC}}$ in the real world. These are adversaries which are under full control of the environment and are only supposed to follow the $\mathcal{E}_{\mathrm{UC}}$'s orders, like injecting messages into the network communication. In other words, the core of the attack resides in the environment $\mathcal{E}_{\mathrm{UC}}$ and the adversary only acts as in interface. Remarkably, if a protocol securely realises a functionality with respect to dummy adversaries, then it also does in general.

The second noteworthy property is delayed outputs in the ideal functionality [Can00]. In the real-world the adversary has full control over the network and may decide when to deliver, say, channel messages. To capture this ability in the ideal model for communication between the ideal functionality and the parties one usually deploys delayed-output instructions, informing the adversary about a message waiting to be delivered to some party. These allow the adversary to decide at which point in time a message eventually gets delivered, by providing a corresponding command to the functionality. However, delayed-output instructions, as defined in [Can00], do not make any stipulation about preserving the order in which messages are delivered. Since this is a crucial characteristic of a secure channel, we enforce such behaviour by associating delayed-output instructions to a queueing mechanism in the ideal functionality.

6.2 Secure Channel Functionality

A secure channel functionality has been given in [CK02]. It consists of a stage in which the channel between two parties P_i and P_j is established. Once this is done, party P_i can securely transmit messages m to the other party. This is performed by sending m to the secure channel functionality. The functionality then informs the adversary about a transmission, but keeps the actual message m secret. Only the length $|m|$ of the message is revealed to the adversary. The adversary can then decide when to deliver the next message to the receiving party P_j.

We adapt this secure channel functionality to the unidirectional setting, i.e., only party P_i sends messages, and it is a single-instance functionality, i.e., it only allows to establish a single channel. The UC composition theorem allows to extend this simple form of a channel to more complex constructions. The resulting secure channel functionality is described in Figure 4.

6.3 Simulatable Channels with Integrity are Universally Composable

Here we show that simulatable channels (with integrity) are also universally composable. The necessity of the integrity property stems from the definition of the ideal channel functionality: The UC adversary can only demand to deliver messages which have been actually inserted into the channel; it cannot make the receiving party output further messages. In contrast, simulatable channels without integrity in principle allow the simulator to output other messages as well. Put differently, the secure channel functionality

Functionality $\mathcal{F}_{\mathrm{SC}}$

1. Upon receiving a command $(\texttt{EstCh}, \mathit{sid}, P_i, P_j)$ from P_i send $(\texttt{EstCh}, \mathit{sid}, P_i, P_j)$ to the adversary and (delayed) to P_j. Store $(\texttt{EstCh}, \mathit{sid}, P_i, P_j)$ and ignore all further establishment requests. Create an empty queue $\mathcal{Q}$ for $(\texttt{EstCh}, \mathit{sid}, P_i, P_j)$

2. Upon receiving a command $(\texttt{Send}, \mathit{sid}, m)$ from P_i check if there is a stored entry $(\texttt{EstCh}, \mathit{sid}, P_i, P_j)$. If not, ignore the message. Else send $(\texttt{Sent}, \mathit{sid}, |m|)$ to the adversary and enqueue $\mathcal{Q}.\mathsf{enq}(m)$ in the queue.

3. Upon receiving a command $(\texttt{Deliver}, \mathit{sid})$ from the adversary, check if $\mathcal{Q}$ is empty; if so, ignore the message. Else dequeue the next message $m \leftarrow \mathcal{Q}.\mathsf{deq}()$ and send $(\texttt{Sent}, \mathit{sid}, m)$ to P_j.

Figure 4: Ideal functionality for a secure channel (with static corruptions).

stipulates integrity by construction.

We are, of course, faced with the problem that the two parties need to share a key in the symmetric setting, without having a way to communicate securely yet. Previous solutions [CK01] assumed that the keys are established by running a suitable key exchange protocol first. To abstract out this step, we design our protocol π_{SC} in the hybrid setting where an ideal functionality $\mathcal{F}_{\mathrm{KE}}$ establishes a shared key between the two parties. That is, π_{SC} may call the ideal functionality $\mathcal{F}_{\mathrm{KE}}$ as part of the protocol steps. We parameterise this functionality by a key generation algorithm $\mathcal{K}$ to describe the underlying distribution over keys. The concrete implementation of the key establishment protocol is a matter of choice, but the UC framework says that any protocol realising $\mathcal{F}_{\mathrm{KE}}$ securely, can then be composed with our protocol π_{SC} to yield a secure, fully implemented protocol for $\mathcal{F}_{\mathrm{SC}}$. We assume that the session identifier sid' of the sub procedure has a one-to-one correspondence with the session identifier sid of the calling protocol, e.g., are given by $\mathit{sid}\|0$ and $\mathit{sid}\|1$.

Functionality $\mathcal{F}_{\mathrm{KE}}^{\mathcal{K}}$

1. Upon receiving a command $(\texttt{EstKey}, \mathit{sid}', P_i, P_j)$ from P_i, check that there is no entry for sid' yet. If so, pick a random key $K \leftarrow \mathcal{K}$ and send $(\texttt{EstKey}, \mathit{sid}', P_i, P_j)$ to the adversary and the (delayed-output) messages $(\texttt{EstKey}, \mathit{sid}', P_i, P_j, K)$ to P_i and P_j.

Figure 5: Ideal functionality for key establishment (with static corruptions).

Construction 6.1 *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Define the protocol π_{SC} in the $\mathcal{F}_{\mathrm{KE}}^{\mathcal{K}}$-hybrid model follows:*

- *On input $(\texttt{EstCh}, \mathit{sid}, P_i, P_j)$ to P_i make a call $(\texttt{EstKey}, \mathit{sid}', P_i, P_j)$ to $\mathcal{F}_{\mathrm{KE}}^{\mathcal{K}}$.*
- *On input $(\texttt{EstKey}, \mathit{sid}', P_i, P_j, K)$ from $\mathcal{F}_{\mathrm{KE}}^{\mathcal{K}}$ to P_i or P_j store $(\mathit{sid}, P_i, P_j, K)$.*
- *On input $(\texttt{Send}, \mathit{sid}, m)$ to P_i check for an entry $(\mathit{sid}, P_i, P_j, K)$. If found, compute $c \leftarrow \mathcal{E}(K, m)$, and possibly update the state, and send (sid, c) to P_j.*
- *On input (sid, f) check for an entry $(\mathit{sid}, P_i, P_j, K)$. If found, compute the sequence $(v_1, m_1), \ldots, (v_\ell, m_\ell) \leftarrow \mathcal{D}(K, f)$, possibly updating the state, and for each $v_i = \top$ output $(\texttt{Sent}, \mathit{sid}, m_i)$ (in this order).*

We state our theorem with respect to the stateful fragmentation notion $\mathsf{FCS\text{-}I}$. The result also transfers straightforwardly to the stateless and stateful atomic cases $\mathsf{CS\text{-}I}$ and $\mathsf{SCS\text{-}I}$.

Theorem 6.2 *If $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ supports fragmentation and is channel simulatable with integrity (*$\mathsf{FCS\text{-}I}$*) then the protocol π_{SC} securely realises $\mathcal{F}_{\text{SC}}$ in the $\mathcal{F}^{\mathcal{K}}_{\text{KE}}$-hybrid model.*

The idea is to turn the channel simulator $\mathcal{S}$, embedded into a wrapper Z, into a UC simulator $\mathcal{S}_{\text{UC}}$, interacting with the channel functionality $\mathcal{F}_{\text{SC}}$ instead. The reduction then shows that any UC environment $\mathcal{E}_{\text{UC}}$ (in combination with a fixed but sufficiently general UC dummy adversary $\tilde{\mathcal{A}}_{\text{UC}}$) against this UC simulator can be transformed into a channel simulatability adversary $\mathcal{A}$. Note that the order of quantifiers is important here: the UC simulator $\mathcal{S}_{\text{UC}}$ works for any environment $\mathcal{E}_{\text{UC}}$ just as the channel simulator $\mathcal{S}$ works for any channel adversary $\mathcal{A}$. Integrity of the channel ensures that the simulation of the UC simulator $\mathcal{S}_{\text{UC}}$ is sound. The proof is in Appendix B.

Unfortunately, we cannot show that universal composability implies channel simulatability (with or without integrity). The reason is that ciphertexts may carry redundancy, e.g., an extra bit appended to the ciphertext $c\|0$, which still allows a UC simulator to detect an altered but valid ciphertext, say, $c\|1$, and to ask the ideal functionality to forward the next message in the queue. Our channel simulator, on the other hand, does not know the message encapsulated in $c\|0$ and the wrapper would not reveal it either.

6.4 Other Work on Composable Secure Channels

In [KT09], Küsters and Tuengerthal consider two ideal functionalities, one for encryption and one for authenticated encryption and present matching protocols which realise these functionalities iff the underlying symmetric encryption schemes respectively satisfy $\mathsf{IND\text{-}CCA}$ and $\mathsf{IND\text{-}CPA} \wedge \mathsf{INT\text{-}CTXT}$. These results are limited to atomic and single-error encryption schemes. More importantly, however, the ideal functionalities considered therein are significantly different from that in [CK02] (and consequently also to ours): They consider the stronger notion of adaptive corruptions and thus have to deal with the committing property of encryption schemes. At the same time, their composition, in an intermediate step, uses an encryption scheme with full key reveals, such that the problem of key cycles —the environment asking for circular encryptions of a key under that key— must be taken care of. In contrast, [CK02] and we here work with the common notion of secret keys.

An alternative formulation of secure channels can be found in [MT10, MRT12], in the language of Maurer's Constructive Cryptography framework. We believe that an analogue of Theorem 6.2 should also hold for the Constuctive Cryptography framework. That is, any scheme that is channel simulatable with integrity ($\mathsf{CS\text{-}I}$/$\mathsf{SCS\text{-}I}$/$\mathsf{FCS\text{-}I}$) can be used to convert an insecure channel into a secure channel.

7 Dropbear's SSH-CTR Implementation is $\mathsf{FCS\text{-}I}$ Secure

Dropbear is an SSH distribution intended specifically for resource-constrained devices such as embedded systems. In a measurement study performed in early 2016 [ADHP16] it was found to be the most widely deployed SSH implementation on the Internet. Owing to its minimalist design it only implements a handful of ciphersuites. Following the attack from [APW09] which affected CBC encryption, it added support for counter mode encryption and set this as the default. The study from [ADHP16] identified counter-mode encryption as the preferred choice for more than 90% of the Dropbear servers.

The $\mathsf{SSH\text{-}CTR}$ scheme described in Figure 6 is an accurate representation of SSH's symmetric encryption using counter mode that we extracted from Dropbear's open source code. Throughout it is assumed that compression is disabled. At various points during decryption a ciphertext may be deemed to be invalid resulting in the connection being torn down. We model this by setting a closed flag at which point all subsequent calls to the decryption algorithm will return an error of the form $(\perp, \mathsf{CONN\_CLOSED})$. Dropbear does not return specific error messages prior to closing a connection, however we adopt a conservative approach and return distinct error messages for every decryption failure that results in a connection tear-down. This only serves to strengthen our security result, since security will hold even if an adversary can distinguish these events through timing information or some other means.

We next show that SSH-CTR is FCS-I secure. To prove this, we need to transform the scheme, through a sequence of game hops, into a pair of algorithms such that a) both algorithm do not make use of the key, b) encryption does not make use of the message contents, and c) decryption only returns error messages for out-of-sync ciphertexts. This is easier than it sounds, in particular by the point where we switch from a block cipher and MAC to their idealised forms (i.e. random functions) we have already eliminated the key. We then only need a couple of simple probabilistic arguments to reach our goal. The advantage of channel simulatability is that we can focus on specific portions of the code without having to worry about its functionality as a whole. For example, we do not have to worry about the parts of the code which handle the reconstruction of ciphertexts and validating of the length field. Indeed if the scheme made use of a nonce-based AEAD scheme, such as GCM, we would only need one game hop to prove channel simulatability.

Below is a formal statement of the security theorem. Its proof can be found in Appendix C.

Theorem 7.1 (SSH-CTR is FCS-I secure) *Let SSH-CTR be the encryption scheme supporting ciphertext fragmentation, composed of a blockcipher BC and a MAC algorithm MAC, described in Figure 6. Then there exists a simulator $\mathcal{S}$ such that for any FCS-I adversary $\mathcal{A}_{\mathsf{fcs}}$ attempting to distinguish $\mathcal{S}$ from SSH-CTR, running in time t, making at most q_e encryption queries totalling μ_e bits, and at most q_d decryption queries totalling μ_d bits, it holds that:*

$$\mathsf{Adv}^{\mathsf{fcs}}_{\mathsf{SSH\text{-}CTR}}(\mathcal{A}_{\mathsf{fcs}}) \leq \mathsf{Adv}^{\mathsf{prp}}_{\mathsf{BC}}(t', q_f) + \frac{q_f^2}{2^{\mathsf{blocksize}+1}} + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{MAC}}(t', q_m) + 2^{-\mathsf{macsize}},$$

where $q_f = \lceil \frac{\mu_e + 40q_e}{\mathsf{blocksize}} \rceil + q_e + \lceil \frac{\mu_d + 40q_d}{\mathsf{blocksize}} \rceil + q_d$, $q_m = q_e + q_d$, and $t' \approx t$.

Furthermore, $\mathcal{S}$ is such that $\overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d}, \cdot)$ never returns an output $(v_1, m'_1), \ldots, (v_\ell, m'_\ell)$ where $v_i = \top$ for some $1 \leq i \leq \ell$.

References

[ABL+14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. (Cited on pages 2, 3, 17, and 18.)

[ADHP16] Martin R. Albrecht, Jean Paul Degabriele, Torben Brandt Hansen, and Kenneth G. Paterson. A surfeit of SSH cipher suites. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16: 23rd Conference on Computer and Communications Security*, pages 1480–1491, Vienna, Austria, October 24–28, 2016. ACM Press. (Cited on pages 2, 3, 4, 5, 6, 7, 18, and 24.)

[AP13] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *2013 IEEE Symposium on Security and Privacy*, pages 526–540, Berkeley, CA, USA, May 19–22, 2013. IEEE Computer Society Press. (Cited on page 3.)

[APW09] 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, Oakland, CA, USA, May 17–20, 2009. IEEE Computer Society Press. (Cited on pages 3 and 24.)

```
alg. $\mathsf{SSH\text{-}CTR}\text{-}\mathcal{E}_K(m)$
parse $K$ as $(K_e, K_m, IV)$
if $\mathsf{e\text{-}seqnr} = 0$
    $\mathsf{e\text{-}ctr} \leftarrow IV$ // initialise on first call
$\mathsf{mlen} \leftarrow |m|_B$
// calculate padding length
$\mathsf{padlen} \leftarrow \mathsf{blocksize} - (5 + \mathsf{mlen})\%\mathsf{blocksize}$
if $\mathsf{padlen} < 4$
    $\mathsf{padlen} \leftarrow \mathsf{padlen} + \mathsf{blocksize}$
// encode the message
$\mathsf{pad} \twoheadleftarrow \{0,1\}^{\mathsf{padlen}\cdot 8}$
$\mathsf{len} \leftarrow 1 + \mathsf{mlen} + \mathsf{padlen}$
$\mathsf{ptxt} \leftarrow \langle \mathsf{len} \rangle_{32} \parallel \langle \mathsf{padlen} \rangle_8 \parallel m \parallel \mathsf{pad}$
// encrypt and mac
$\tau \leftarrow \mathsf{MAC}(K_m, \langle \mathsf{e\text{-}seqnr} \rangle_{32} \parallel \mathsf{ptxt})$
$z \leftarrow \varepsilon$
while $|z| < |\mathsf{ptxt}|$
    $z \leftarrow z \parallel \mathsf{BC}(K_e, \mathsf{e\text{-}ctr})$
    $\mathsf{e\text{-}ctr} \leftarrow \mathsf{e\text{-}ctr} + 1$
$c \leftarrow (\mathsf{ptxt} \oplus z) \parallel \tau$
$\mathsf{e\text{-}seqnr} \leftarrow \mathsf{e\text{-}seqnr} + 1$
return $c$
```

```
alg. $\mathsf{SSH\text{-}CTR}\text{-}\mathcal{D}_K(f)$
parse $K$ as $(K_e, K_m, IV)$
if $\mathsf{d\text{-}seqnr} = 0 \wedge \alpha = \varepsilon$
    $\mathsf{d\text{-}ctr} \leftarrow IV$ // initialise on first call
if $\mathsf{closed}$
    $\mathsf{out} \leftarrow (\perp, \mathsf{CONN\_CLOSED})$; break
$\alpha \leftarrow \alpha \parallel f$; $\mathsf{out} \leftarrow \varepsilon$ // update buffer and reset output
while $(\mathsf{true})$ // process buffer ($\alpha$)
    if $|\alpha|_B < \mathsf{blocksize}$
        break // first ciphertext block is incomplete
    // decrypt first ciphertext block
    $\mathsf{ptxt}' \leftarrow \alpha[1, \mathsf{blocksize}] \oplus \mathsf{BC}(K_e, \mathsf{d\text{-}ctr})$
    $\mathsf{d\text{-}ctr} \leftarrow \mathsf{d\text{-}ctr} + 1$
    $\mathsf{clen} \leftarrow \langle \mathsf{ptxt}'[1, 32] \rangle^{-1} + 4 + \mathsf{macsize}$
    $\mathsf{inRange} \leftarrow (16 + \mathsf{macsize} \le \mathsf{clen} \le 35000)$
    $\mathsf{isMult} \leftarrow ((\mathsf{clen} - \mathsf{macsize})\%\mathsf{blocksize} \neq 0)$
    if $\neg\, \mathsf{inRange} \vee \mathsf{isMult}$ // validate length
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_LENGTH})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    if $|\alpha|_B < \mathsf{clen}$
        break // wait to complete ciphertext
    $z \leftarrow \varepsilon$ // decrypt and verify mac
    while $|z| < (\mathsf{clen} - \mathsf{blocksize} - \mathsf{macsize})$
        $z \leftarrow z \parallel \mathsf{BC}(K_e, \mathsf{e\text{-}ctr})$
        $\mathsf{d\text{-}ctr} \leftarrow \mathsf{d\text{-}ctr} + 1$
    $z \leftarrow z[1, \mathsf{clen} - \mathsf{blocksize} - \mathsf{macsize}]$ // trim
    $\mathsf{ptxt}' \leftarrow \mathsf{ptxt}' \parallel z \oplus \alpha[\mathsf{blocksize} + 1, \mathsf{clen} - \mathsf{macsize}]_B$
    $\tau' \leftarrow \alpha[\mathsf{clen} - \mathsf{macsize} + 1, \mathsf{clen}]_B$
    $\alpha \leftarrow \alpha[\mathsf{clen} + 1, *]_B$ // remove decrypted ciphertext
    if $\tau' \neq \mathsf{MAC}(K_m, \langle \mathsf{d\text{-}seqnr} \rangle_{32} \parallel \mathsf{ptxt}')$
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_MAC})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    $\mathsf{padlen} \leftarrow \langle \mathsf{ptxt}'[5, 5]_B \rangle^{-1}$ // validate padding length
    $\mathsf{mlen}' \leftarrow \mathsf{clen} - \mathsf{padlen} - 4 - 1 - \mathsf{macsize}$
    if $(\mathsf{mlen}' > 32789) \vee (\mathsf{mlen}' < 1)$
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_PAD\_LENGTH})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    $m' \leftarrow \mathsf{ptxt}'[6, \mathsf{clen} - \mathsf{macsize} - \mathsf{padlen}]_B$
    $\mathsf{out} \leftarrow \mathsf{out} \parallel (\top, m')$
    $\mathsf{d\text{-}seqnr} \leftarrow \mathsf{d\text{-}seqnr} + 1$
return $\mathsf{out}$
```

Figure 6: The SSH-CTR scheme as implemented in Dropbear.

[AR07] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 20(3):395, July 2007. (Cited on pages 1 and 9.)

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. (Cited on page 9.)

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. (Cited on pages 1 and 5.)

[BDPS12] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. Security of symmetric encryption in the presence of ciphertext fragmentation. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 682–699, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. (Cited on pages 2, 3, 4, 5, 6, 7, and 18.)

[BDPS14] 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 *Lecture Notes in Computer Science*, pages 367–390, Singapore, March 11–13, 2014. Springer, Heidelberg, Germany. (Cited on pages 2, 3, and 18.)

[BKN02] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In Vijayalakshmi Atluri, editor, *ACM CCS 02: 9th Conference on Computer and Communications Security*, pages 1–11, Washington D.C., USA, November 18–22, 2002. ACM Press. (Cited on pages 2, 3, 5, 7, and 18.)

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany. (Cited on page 7.)

[BPS15] Guy Barwell, Daniel Page, and Martijn Stam. Rogue decryption failures: Reconciling AE robustness notions. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *Lecture Notes in Computer Science*, pages 94–111, Oxford, UK, December 15–17, 2015. Springer, Heidelberg, Germany. (Cited on pages 2, 3, 5, 7, 12, and 18.)

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/2000/067`. (Cited on pages 21 and 30.)

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. (Cited on page 21.)

[CHVV03] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. (Cited on page 3.)

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany. (Cited on page 22.)

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. (Cited on pages 22, 23, and 24.)

[Des00] Anand Desai. New paradigms for constructing symmetric encryption schemes secure against chosen-ciphertext attack. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 394–412, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany. (Cited on pages 11 and 13.)

[DP10] 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, Chicago, Illinois, USA, October 4–8, 2010. ACM Press. (Cited on page 3.)

[FGMP15] Marc Fischlin, Felix Günther, Giorgia Azzurra Marson, and Kenneth G. Paterson. Data is a stream: Security of stream-based channels. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 545–564, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. (Cited on pages 2, 4, 7, 12, and 18.)

[Fis99] Marc Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 432–445, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. (Cited on page 9.)

[HKR15] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 15–44, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. (Cited on pages 2, 3, 17, and 18.)

[HRRV15] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online authenticated-encryption and its nonce-reuse misuse-resistance. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 493–517, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. (Cited on page 6.)

[KT09] Ralf Küsters and Max Tuengerthal. Universally composable symmetric encryption. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium, CSF 2009, Port Jefferson, New York, USA, July 8-10, 2009*, pages 293–307. IEEE Computer Society, 2009. (Cited on page 23.)

[MRT12] Ueli Maurer, Andreas Rüedlinger, and Björn Tackmann. Confidentiality and integrity: A constructive perspective. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 209–229, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany. (Cited on page 24.)

[MT10] Ueli Maurer and Björn Tackmann. On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 505–515, Chicago, Illinois, USA, October 4–8, 2010. ACM Press. (Cited on page 24.)

[PW10] 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 *Lecture Notes in Computer Science*, pages 345–361, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. (Cited on pages 2, 3, and 18.)

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 196–205, Philadelphia, PA, USA, November 5–8, 2001. ACM Press. (Cited on page 1.)

[Rog04] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359, New Delhi, India, February 5–7, 2004. Springer, Heidelberg, Germany. (Cited on page 1.)

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. (Cited on pages 2 and 18.)

A Proof of Theorem 5.4

Noting that $\mathsf{cfDec}(\cdot)$ is identical to $\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)$, we then have that for any channel simulator $\mathcal{S}$ and any adversary $\mathcal{A}$ with at most $\mathcal{R}_\mathcal{A}$ resources:

$$\underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\,,\mathsf{cfDec}(\cdot)\\ \mathcal{S}(\mathsf{e},\cdot),\overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d},\cdot)\end{bmatrix} = \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\,,\overline{\mathsf{Z}}[\mathcal{D}_K](\cdot)\\ \mathcal{S}(\mathsf{e},\cdot),\overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d},\cdot)\end{bmatrix}.$$

Then, by a straightforward reduction that applies $\overline{\mathsf{Z}}$ to the decryption oracle and observing that $\overline{\mathsf{Z}}[\mathsf{Z}[\mathcal{S}]](\mathsf{d},\cdot)$ is identical to $\overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d},\cdot)$, it follows that:

$$\begin{aligned}&\leq \underset{\mathcal{A}}{\Delta}\begin{bmatrix}\mathcal{E}_K(\cdot)\,,\mathcal{D}_K(\cdot)\\ \mathcal{S}(\mathsf{e},\cdot),\mathsf{Z}[\mathcal{S}](\cdot)\end{bmatrix},\\ &= \quad \mathsf{Adv}^{\mathsf{fcs}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}).\end{aligned}$$

From the above relation it then follows that:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{int\text{-}sfcfrg}}_{\mathcal{SE}}(\mathcal{A}) &= \Pr\Big[\,K \twoheadleftarrow \mathcal{K},\, \mathcal{A}^{\mathcal{E}_K(\cdot),\mathsf{cfDec}(\cdot)} : \mathsf{FORGE}\,\Big],\\ &\leq \Pr\Big[\,\mathcal{A}^{\mathcal{S}(\mathsf{e},\cdot),\overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d},\cdot)} : \mathsf{FORGE}\,\Big] + \mathsf{Adv}^{\mathsf{fcs}}_{\mathcal{SE}}(\mathcal{A},\mathcal{S}).\end{aligned}$$

Now since $\mathcal{SE}$ is $(\epsilon, \mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{A}})$-$\mathsf{FCS\text{-}I}$ secure, there exists a simulator such that the first term is zero and the second term is bounded by ϵ, thus:

$$\leq \epsilon.$$

$\square$

B Simulatable Channels with Integrity are Universally Composable

B.1 Technical Details

Before proving UC security of channel simulatable protocols with integrity we fill in a few more details.

In principle, the adversary may decide to corrupt parties, either at the outset of the execution (static corruptions) or while the execution is running (adaptive corruptions). This choice affects even the design of the ideal functionality. However, for channels based on symmetric keys, corrupting one of the participating parties discloses the secret key and the channel can no longer provide any security guarantees. Hence, we assume that the adversary neither corrupts the sender nor the receiver in our scenario.

Some technicalities of the UC framework are relevant to our setting. The first is that it suffices to consider so-called dummy adversaries $\tilde{\mathcal{A}}_{\mathrm{UC}}$ in the real world. These are adversaries which are under full control of the environment and are only supposed to follow the $\mathcal{E}_{\mathrm{UC}}$'s orders, like injecting messages into the network communication. In other words, the core of the attack resides in the environment $\mathcal{E}_{\mathrm{UC}}$ and the adversary only acts as in interface. Remarkably, if a protocol securely realises a functionality with respect to dummy adversaries, then it also does in general.

The second noteworthy property is delayed outputs in the ideal functionality [Can00]. In the real-world the adversary has full control over the network and may decide when to deliver, say, channel messages. To capture this ability in the ideal model for communication between the ideal functionality and the parties one usually deploys delayed-output instructions, informing the adversary about a message waiting to be delivered to some party. These allow the adversary to decide at which point in time a message eventually gets delivered, by providing a corresponding command to the functionality. However, delayed-output instructions, as defined in [Can00], do not make any stipulation about preserving the order in which messages are delivered. Since this is a crucial characteristic of a secure channel, we enforce such behaviour by associating delayed-output instructions to a queueing mechanism in the ideal functionality.

B.2 Proof of Theorem 6.2

Proof. The UC simulator is essentially given by the simulator for channel simulatability. More precisely, simulator $\mathcal{S}_{\mathrm{UC}}$ runs an internal copy of the dummy real-world adversary $\tilde{\mathcal{A}}_{\mathrm{UC}}$ with which it relays the communication with the environment, e.g., if the environment asks the adversary to report the network communication. Our simulator will also run an internal copy of the encryption simulator $\mathcal{S}$, also maintaining its state, and for each session a pruned transcript T_c which only stores ciphertexts (instead of message-ciphertext pairs). It initialises the variable sync to true and sequences F and C to empty. The simulator $\mathcal{S}_{\mathrm{UC}}$ operates as follows.

- If the adversary $\tilde{\mathcal{A}}_{\mathrm{UC}}$, at the beginning, decides to corrupt either party (different from the sender and receiver) then $\mathcal{S}_{\mathrm{UC}}$ corrupts the same party.

- If the simulator is informed of a message $(\mathtt{EstCh}, \mathit{sid}, P_i, P_j)$ by the functionality $\mathcal{F}_{\mathrm{SC}}$ it internally stores (sid, P_i, P_j).

- If the simulator is informed of a message $(\texttt{Sent}, \mathit{sid}, \ell)$ by the functionality, then it calls the encryption simulator $\mathcal{S}(\mathsf{e}, \ell)$ to create a ciphertext c (where the simulator $\mathcal{S}$ possibly updates its state). The UC simulator $\mathcal{S}_{\mathrm{UC}}$ appends c to T_c and returns (sid, c) to its internal copy of $\tilde{\mathcal{A}}_{\mathrm{UC}}$.

- If the adversary sends a ciphertext fragment (sid, f') to a party P_j in the emulation, and $\mathcal{S}_{\mathrm{UC}}$ has indeed stored (sid, P_i, P_j), then the simulator $\mathcal{S}_{\mathrm{UC}}$ emulates the steps of $\mathsf{Z}_{\mathcal{S}}(f')$ (without knowing the enqueued messages in T_c, though). That is, it first calls $(v_1, m'_1), \ldots, (v_\ell, m'_\ell) \leftarrow \mathcal{S}(\mathsf{d}, f')$ to update the internal state of the simulator, and updates $F \leftarrow F \,\|\, f'$. If sync is false then it ignores this message. If T_c is empty then it sets $\mathsf{sync} \leftarrow \mathsf{false}$ and ignores the messages. Otherwise it repeatedly computes $c \leftarrow \mathsf{T}_c.\mathsf{next}()$, updates $C \leftarrow C \,\|\, c$. As long as C is still a prefix of F then it will permit delivery of the waiting message $(\texttt{Sent}, \mathit{sid}, m)$ to P_j by forwarding $(\texttt{Deliver}, \mathit{sid})$ to the ideal functionality (where the simulator $\mathcal{S}_{\mathrm{UC}}$ does not get to see m). If it reaches an out-of-sync ciphertext it sets sync to false and from then on ignores all incoming messages (sid, f') to P_j.

An attack by an adversary $\tilde{\mathcal{A}}_{\mathrm{UC}}$, instructed by the environment $\mathcal{E}_{\mathrm{UC}}$, can be turned into an adversary $\mathcal{A}$ against channel simulatability under fragmentation (with integrity). That is, given an environment $\mathcal{E}_{\mathrm{UC}}$ (and the dummy adversary $\tilde{\mathcal{A}}_{\mathrm{UC}}$) we can construct adversary $\mathcal{A}$ as follows.

- If $\mathcal{E}_{\mathrm{UC}}$ never establishes a channel and all messages are ignored, then the adversary can trivially emulate the attack by doing nothing.

- For established channels, for any input $(\texttt{Send}, \mathit{sid}, m)$ of $\mathcal{E}_{\mathrm{UC}}$ to party P_i, adversary $\mathcal{A}$ calls its encryption oracle interface ($\mathcal{E}(K, \cdot)$ or $\mathcal{S}(\mathsf{e}, |\cdot|)$) about m to get c and returns (sid, c) to $\mathcal{E}_{\mathrm{UC}}$. It also stores (m, c) in a list T.

- For any network message (sid, f') for established channels, sent upon request of $\mathcal{E}_{\mathrm{UC}}$ to P_j, adversary $\mathcal{A}$ calls its decryption oracle interface ($\mathcal{D}(K, \cdot)$ or $\mathcal{S}(\mathsf{d}, \cdot)$) about f' and returns $(\texttt{Sent}, \mathit{sid}, m_i)$, in this order, for any replies $(v_1, m_1), \ldots, (v_\ell, m_\ell)$ with $v_i = \top$.

- Finally, $\mathcal{A}$ returns the environment's output bit.

Assume that the environment $\mathcal{E}_{\mathrm{UC}}$ can successfully tell apart the real world setting from the ideal world scenario. We claim that, by the construction above, this translates into a successful adversary $\mathcal{A}$ against channel simulatability. First note that, if $\mathcal{A}$ interacts with the encryption and decryption interfaces $\mathcal{E}(K, \cdot)$ and $\mathcal{D}(K, \cdot)$ for a randomly chosen key $K \leftarrow \mathcal{K}$, then $\mathcal{A}$ perfectly emulates an attack of $\mathcal{E}_{\mathrm{UC}}$ against the real protocol π_{SC} (in the $\mathcal{F}_{\mathrm{KE}}$-hybrid model). The reason is that the key in π_{SC} then has the same distribution as the sending and receiving interfaces. It follows that $\mathcal{A}$'s oracles, together with the recovery strategy, perfectly emulate the real-world protocol π_{SC}.

If, on the other hand, adversary $\mathcal{A}$ interacts with the simulated channel interfaces $\mathcal{S}(\mathsf{e}, |\cdot|)$ and $\mathcal{S}(\mathsf{d}, \cdot)$, then this corresponds to the environment $\mathcal{E}_{\mathrm{UC}}$ communicating with the ideal world and the simulator $\mathcal{S}_{\mathrm{UC}}$. There are two differences, though. First, in the simulation through $\mathcal{S}_{\mathrm{UC}}$ we use the storage system of the ideal functionality to deliver the next message m_i, whereas $\mathsf{Z}[\mathcal{S}]$ in $\mathcal{A}$'s calls uses its internal message-ciphertext list to restore the message. But this is only a syntactic difference. The other divergence lies in the way how out-of-order ciphertexts c' in f' are treated: The UC simulator $\mathcal{S}_{\mathrm{UC}}$ will ignore such ciphertexts right away; the adversary $\mathcal{A}$ would first receive a reply (v, m') from its simulator's decryption interface. But by the channel integrity this can only result in $v = \bot$ such that the adversary $\mathcal{A}$, too, ignores this ciphertext. It follows that the derived channel adversary $\mathcal{A}$ from the environment $\mathcal{E}_{\mathrm{UC}}$ perfectly emulates the ideal-world attack when having access to the channel simulator $\mathcal{S}$.

Note that the channel simulator $\mathcal{S}$ works for any adversary $\mathcal{A}$. Hence, our UC simulalator $\mathcal{S}_{\mathrm{UC}}$, which is fully determined by the scheme, the simulator $\mathcal{S}$ and the dummy adversary $\tilde{\mathcal{A}}_{\mathrm{UC}}$, works for any environment $\mathcal{E}_{\mathrm{UC}}$. This means that the protocol π_{SC} securely realises the functionality $\mathcal{F}_{\mathrm{SC}}$ in the $\mathcal{F}_{\mathrm{KE}}$-hybrid model. □

C Proof of Theorem 7.1

We prove the theorem through a sequence of games, where we gradually morph the encryption and decryption algorithms into the desired simulator. Throughout all games, the decryption algorithm is augmented with the wrapper Z described in Figure 3. Let $\mathcal{A}_{\mathsf{fcs}}$ denote the $\mathsf{FCS\text{-}I}$ adversary and assume that its runtime is bounded by t, it makes at most q_e encryption queries totalling μ_e bits, and at most q_d decryption queries totalling μ_d bits.

G0 In this game the adversary is given oracle access to $\mathsf{SSH\text{-}CTR\text{-}}\mathcal{E}_K(\cdot)$ and $\mathsf{Z}[\mathsf{SSH\text{-}CTR\text{-}}\mathcal{D}_K](\cdot)$. The descriptions of $\mathsf{SSH\text{-}CTR\text{-}}\mathcal{E}_K(\cdot)$ and $\mathsf{SSH\text{-}CTR\text{-}}\mathcal{D}_K(\cdot)$ can be found in Figure 6.

G1 We now replace BC and MAC with matching random functions $\mathsf{R_E}$ and $\mathsf{R_M}$. Then for any adversary $\mathcal{A}_{\mathsf{fcs}}$ we have that:

$$\Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G0}} \Rightarrow 1\right] \leq \mathsf{Adv}_{\mathsf{BC}}^{\mathsf{prf}}(t', q_f) + \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{prf}}(t', q_m),$$

where $q_f = \lceil \frac{\mu_e + 40 q_e}{\mathsf{blocksize}} \rceil + q_e + \lceil \frac{\mu_d + 40 q_d}{\mathsf{blocksize}} \rceil + q_d$, $q_m = q_e + q_d$, and $t' \approx t$. By further applying the PRP/PRF switching lemma we then obtain:

$$\Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G0}} \Rightarrow 1\right] \leq \mathsf{Adv}_{\mathsf{BC}}^{\mathsf{prp}}(t', q_f) + \frac{q_f^2}{2^{\mathsf{blocksize}+1}} + \mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{prf}}(t', q_m), \quad (1)$$

In addition we dispose of the keys (as they are no longer needed) and set the global variable IV to a random value upon the adversary's first query. Note that this modification does not affect the behaviour of any of the two oracles and hence the above inequality remains valid.

G2 In this game we introduce a list C that will be shared by the encryption and decryption oracles. Now we modify the encryption oracle so that before outputting a ciphertext it will append it to this list. At the decryption side, we replace line 26 in Figure 6 with:

$$\begin{array}{l} \hat{c} \leftarrow \alpha[1, \mathsf{clen} - \mathsf{macsize}]_B \\ \mathsf{ptxt}' \leftarrow \mathsf{ptxt}' \parallel (z \oplus \hat{c}[\mathsf{blocksize}+1, *]) \end{array}$$

and the if condition at line 29 with:

$$\textbf{if } \hat{c} \parallel \tau' \neq \mathsf{C.next}()$$

The change at line 26 simply introduces the variable $\hat{c}$ without affecting the functionality. The change at line 29 will now cause the decryption oracle to return $(\bot, \mathsf{INVALID\_MAC})$ and set the closed flag as soon as it detects that the reconstructed ciphertext has been modified by the adversary. On the other hand as long as the adversary remains passive, simply forwarding and fragmenting ciphertexts without altering them, the decryption oracle's behaviour is the same as in the previous game. Thus the two games are equivalent until the following condition (bad event) occurs:

$$\hat{c} \parallel \tau' \neq \mathsf{C.next}() \wedge \tau' = \mathsf{R_M}(\langle \mathsf{d\text{-}seqnr} \rangle_{32} \parallel \mathsf{ptxt}').$$

Now, to calculate the probability of this event there are two cases to consider:

a. The $\hat{c}$ part has been modified. This implies that ptxt' is different and consequently that $\mathsf{R_M}()$ is called on a fresh input. Then τ' is independent of $\mathsf{R_M}(\langle \mathsf{d\text{-}seqnr}\rangle_{32} \parallel \mathsf{ptxt}')$ and the probability of the two values being equal is $2^{-\mathsf{macsize}}$.

b. The $\hat{c}$ part is unchanged but τ' has been modified. Since the $\hat{c}$ is unchanged then also ptxt' is unchanged, and it follows that the probability that $\tau' = \mathsf{R_M}(\langle \mathsf{d\text{-}seqnr}\rangle_{32} \parallel \mathsf{ptxt}')$ is zero.

Thus we have that:

$$\Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G2}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G1}} \Rightarrow 1\right] \le 2^{-\mathsf{macsize}}. \tag{2}$$

At this point the encryption and decryption algorithms look as described in Figure 7.

G3 In this game we make two changes which leave the overall behaviour of the game unchanged. The first alteration is to defer plaintext recovery during decryption to the point right before padding-length validation, which is where it is first needed. This is possible because decryption no longer needs to compute a MAC over the plaintext. More precisely, we move lines 21 to 25 and line 27 in Figure 7, without changing their order, to the point just preceding line 33 in that same figure.

As for the second modification, we introduce an additional random function R'_{E}, whose output is 32 bits long, which will be used solely for the purpose of encrypting and decrypting the length field. Specifically, during encryption the counter value that is input to $\mathsf{R_E}$ for processing the first plaintext block is also fed to R'_{E}, and the first 32 bits of the output from $\mathsf{R_E}$ are replaced with the output from R'_{E}. This is handled with the addition of lines 17 and 21 in Figure 8. As for decryption, we change line 11 of Figure 7 to only decrypt the length field instead of the whole first block, using R'_{E}, and without incrementing the counter $\mathsf{d\text{-}ctr}$. The decryption of the remaining part of the first block is now handled when decrypting the remaining blocks in the highlighted code starting at line 26 of Figure 8. From the above it follows that:

$$\Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G3}} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G2}} \Rightarrow 1\right], \tag{3}$$

G4 We now make our final alteration which yields the desired simulator, described in Figure 8. We change the encryption oracle to take as input only the length of the message len and internally set m to 0^{len} (line 4). This change is possible because now the output of the decryption algorithm in combination with the wrapper is independent of $\mathsf{R_E}$. To see this, note that while the adversary is passive, and hence the condition on line 23 of Figure 8 always evaluates to false, the output of the decryption algorithm is overwritten by Z, which is clearly independent of $\mathsf{R_E}$. On the other hand as soon as the adversary becomes active, the closed flag is set and lines 26 to 39 (which is where $\mathsf{R_E}$ is used) are never executed. Accordingly, the distribution of the ciphertexts returned by the encryption oracle is unchanged from the adversary's point of view. Hence:

$$\Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G4}} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_{\mathsf{fcs}}^{\mathsf{G3}} \Rightarrow 1\right], \tag{4}$$

The algorithms described in G4 (Figure 8) now satisfy the required structure for the simulator. Combining (1), (2), (3), and, (4), yields the bound in Theorem 7.1. Furthermore, from lines 23-25 and 4-5 of the decryption simulator it follows that $\overline{\mathsf{Z}}[\mathcal{S}](\mathsf{d},\cdot)$ never returns an output other than $(\perp, \mathsf{INVALID\_MAC})$, and therefore the integrity condition is also satisfied. $\square$

```
alg. SSH-CTR-$\mathcal{E}(m)$

if $IV = \varepsilon$ then $IV \twoheadleftarrow \{0,1\}^{\mathsf{blocksize}}$
if $\mathsf{e\text{-}seqnr} = 0$
    $\mathsf{e\text{-}ctr} \leftarrow IV$ // initialise on first call
$\mathsf{mlen} \leftarrow |m|_B$
// calculate padding length
$\mathsf{padlen} \leftarrow \mathsf{blocksize} - (5 + \mathsf{mlen})\%\mathsf{blocksize}$
if $\mathsf{padlen} < 4$
    $\mathsf{padlen} \leftarrow \mathsf{padlen} + \mathsf{blocksize}$
// encode the message
$\mathsf{pad} \twoheadleftarrow \{0,1\}^{\mathsf{padlen} \cdot 8}$
$\mathsf{len} \leftarrow 1 + \mathsf{mlen} + \mathsf{padlen}$
$\mathsf{ptxt} \leftarrow \langle \mathsf{len} \rangle_{32} \parallel \langle \mathsf{padlen} \rangle_8 \parallel m \parallel \mathsf{pad}$
// encrypt and mac
$\tau \leftarrow \mathsf{R_M}(\langle \mathsf{e\text{-}seqnr} \rangle_{32} \parallel \mathsf{ptxt})$
$z \leftarrow \varepsilon$
while $|z| < |\mathsf{ptxt}|$
    $z \leftarrow z \parallel \mathsf{R_E}(\mathsf{e\text{-}ctr})$
    $\mathsf{e\text{-}ctr} \leftarrow \mathsf{e\text{-}ctr} + 1$
$c \leftarrow (\mathsf{ptxt} \oplus z) \parallel \tau$
$\mathsf{e\text{-}seqnr} \leftarrow \mathsf{e\text{-}seqnr} + 1$
$\mathsf{C.append}(c)$
return $c$
```

```
alg. SSH-CTR-$\mathcal{D}(f)$

if $IV = \varepsilon$ then $IV \twoheadleftarrow \{0,1\}^{\mathsf{blocksize}}$
if $\mathsf{d\text{-}seqnr} = 0 \wedge \alpha = \varepsilon$
    $\mathsf{d\text{-}ctr} \leftarrow IV$ // initialise on first call
if closed
    $\mathsf{out} \leftarrow (\perp, \mathsf{CONN\_CLOSED})$; break
$\alpha \leftarrow \alpha \parallel f; \mathsf{out} \leftarrow \varepsilon$ // update buffer and reset output
while (true) // process buffer ($\alpha$)
    if $|\alpha|_B < \mathsf{blocksize}$
        break // first ciphertext block is incomplete
    // decrypt first ciphertext block
    $\mathsf{ptxt}' \leftarrow \alpha[1, \mathsf{blocksize}] \oplus \mathsf{R_E}(\mathsf{d\text{-}ctr})$
    $\mathsf{d\text{-}ctr} \leftarrow \mathsf{d\text{-}ctr} + 1$
    $\mathsf{clen} \leftarrow \langle \mathsf{ptxt}'[1, 32] \rangle^{-1} + 4 + \mathsf{macsize}$
    $\mathsf{inRange} \leftarrow (16 + \mathsf{macsize} \leq \mathsf{clen} \leq 35000)$
    $\mathsf{isMult} \leftarrow ((\mathsf{clen} - \mathsf{macsize})\%\mathsf{blocksize} \neq 0)$
    if $\neg\, \mathsf{inRange} \vee \mathsf{isMult}$ // validate length
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_LENGTH})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    if $|\alpha|_B < \mathsf{clen}$
        break // wait to complete ciphertext
    $z \leftarrow \varepsilon$ // decrypt and verify mac
    while $|z| < (\mathsf{clen} - \mathsf{blocksize} - \mathsf{macsize})$
        $z \leftarrow z \parallel \mathsf{R_E}(\mathsf{d\text{-}ctr})$
        $\mathsf{d\text{-}ctr} \leftarrow \mathsf{d\text{-}ctr} + 1$
    $z \leftarrow z[1, \mathsf{clen} - \mathsf{blocksize} - \mathsf{macsize}]$ // trim
    $\hat{c} \leftarrow \alpha[1, \mathsf{clen} - \mathsf{macsize}]_B$
    $\mathsf{ptxt}' \leftarrow \mathsf{ptxt}' \parallel (z \oplus \hat{c}[\mathsf{blocksize} + 1, *])$
    $\tau' \leftarrow \alpha[\mathsf{clen} - \mathsf{macsize} + 1, \mathsf{clen}]_B$
    $\alpha \leftarrow \alpha[\mathsf{clen} + 1, *]_B$ // remove decrypted ciphertext
    if $\hat{c} \parallel \tau' \neq \mathsf{C.next}()$
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_MAC})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    $\mathsf{padlen} \leftarrow \langle \mathsf{ptxt}'[5, 5]_B \rangle^{-1}$ // validate padding length
    $\mathsf{mlen}' \leftarrow \mathsf{clen} - \mathsf{padlen} - 4 - 1 - \mathsf{macsize}$
    if $(\mathsf{mlen}' > 32789) \vee (\mathsf{mlen}' < 1)$
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_PAD\_LENGTH})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    $m' \leftarrow \mathsf{ptxt}'[6, \mathsf{clen} - \mathsf{macsize} - \mathsf{padlen}]_B$
    $\mathsf{out} \leftarrow \mathsf{out} \parallel (\top, m')$
    $\mathsf{d\text{-}seqnr} \leftarrow \mathsf{d\text{-}seqnr} + 1$
return out
```

Figure 7: The encryption and decryption algorithms used in G2. The highlighted code indicates the changes with respect to G0.

```
alg. $\mathcal{S}(\mathsf{e}, len)$
if $IV = \varepsilon$ then $IV \twoheadleftarrow \{0,1\}^{\mathsf{blocksize}}$
if $\mathsf{e\text{-}seqnr} = 0$
    $\mathsf{e\text{-}ctr} \leftarrow IV$ // initialise on first call
$m \leftarrow 0^{len}$
$\mathsf{mlen} \leftarrow |m|_B$
// calculate padding length
$\mathsf{padlen} \leftarrow \mathsf{blocksize} - (5 + \mathsf{mlen})\%\mathsf{blocksize}$
if $\mathsf{padlen} < 4$
    $\mathsf{padlen} \leftarrow \mathsf{padlen} + \mathsf{blocksize}$
// encode the message
$\mathsf{pad} \twoheadleftarrow \{0,1\}^{\mathsf{padlen} \cdot 8}$
$\mathsf{len} \leftarrow 1 + \mathsf{mlen} + \mathsf{padlen}$
$\mathsf{ptxt} \leftarrow \langle \mathsf{len} \rangle_{32} \parallel \langle \mathsf{padlen} \rangle_8 \parallel m \parallel \mathsf{pad}$
// encrypt and mac
$\tau \leftarrow \mathsf{R_M}(\langle \mathsf{e\text{-}seqnr} \rangle_{32} \parallel \mathsf{ptxt})$
$z \leftarrow \varepsilon$
$z' \leftarrow \mathsf{R'_E}(\mathsf{e\text{-}ctr})$
while $|z| < |\mathsf{ptxt}|$
    $z \leftarrow z \parallel \mathsf{R_E}(\mathsf{e\text{-}ctr})$
    $\mathsf{e\text{-}ctr} \leftarrow \mathsf{e\text{-}ctr} + 1$
$z[1, 32] \leftarrow z'$
$c \leftarrow (\mathsf{ptxt} \oplus z) \parallel \tau$
$\mathsf{e\text{-}seqnr} \leftarrow \mathsf{e\text{-}seqnr} + 1$
$\mathsf{C.append}(c)$
return $c$
```

```
alg. $\mathcal{S}(\mathsf{d}, f)$
if $IV = \varepsilon$ then $IV \twoheadleftarrow \{0,1\}^{\mathsf{blocksize}}$
if $\mathsf{d\text{-}seqnr} = 0 \wedge \alpha = \varepsilon$
    $\mathsf{d\text{-}ctr} \leftarrow IV$ // initialise on first call
if closed
    $\mathsf{out} \leftarrow (\perp, \mathsf{CONN\_CLOSED})$; break
$\alpha \leftarrow \alpha \parallel f$; $\mathsf{out} \leftarrow \varepsilon$ // update buffer and reset output
while (true) // process buffer ($\alpha$)
    if $|\alpha|_B < \mathsf{blocksize}$
        break // first ciphertext block is incomplete
    // decrypt first ciphertext block
    $\mathsf{ptxt}' \leftarrow \alpha[1, 32] \oplus \mathsf{R'_E}(\mathsf{d\text{-}ctr})$
    $\mathsf{clen} \leftarrow \langle \mathsf{ptxt}'[1, 32] \rangle^{-1} + 4 + \mathsf{macsize}$
    $\mathsf{inRange} \leftarrow (16 + \mathsf{macsize} \leq \mathsf{clen} \leq 35000)$
    $\mathsf{isMult} \leftarrow ((\mathsf{clen} - \mathsf{macsize})\%\mathsf{blocksize} \neq 0)$
    if $\neg\,\mathsf{inRange} \vee \mathsf{isMult}$ // validate length
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_LENGTH})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    if $|\alpha|_B < \mathsf{clen}$
        break // wait to complete ciphertext
    $\hat{c} \leftarrow \alpha[1, \mathsf{clen} - \mathsf{macsize}]_B$
    $\tau' \leftarrow \alpha[\mathsf{clen} - \mathsf{macsize} + 1, \mathsf{clen}]_B$
    $\alpha \leftarrow \alpha[\mathsf{clen} + 1, *]_B$ // remove extracted ciphertext
    if $\hat{c} \parallel \tau' \neq \mathsf{C.next()}$
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_MAC})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    $z \leftarrow \varepsilon$ // recover plaintext
    while $|z| < (\mathsf{clen} - \mathsf{macsize})$
        $z \leftarrow z \parallel \mathsf{R_E}(\mathsf{d\text{-}ctr})$
        $\mathsf{d\text{-}ctr} \leftarrow \mathsf{d\text{-}ctr} + 1$
    $z \leftarrow z[33, \mathsf{clen} - \mathsf{macsize}]$ // trim
    $\mathsf{ptxt}' \leftarrow \mathsf{ptxt}' \parallel z \oplus \hat{c}[33, *]$
    $\mathsf{padlen} \leftarrow \langle \mathsf{ptxt}'[5, 5]_B \rangle^{-1}$ // validate padding length
    $\mathsf{mlen}' \leftarrow \mathsf{clen} - \mathsf{padlen} - 4 - 1 - \mathsf{macsize}$
    if $(\mathsf{mlen}' > 32789) \vee (\mathsf{mlen}' < 1)$
        $\mathsf{out} \leftarrow \mathsf{out} \parallel (\perp, \mathsf{INVALID\_PAD\_LENGTH})$
        $\mathsf{closed} \leftarrow \mathsf{true}$; break
    $m' \leftarrow \mathsf{ptxt}'[6, \mathsf{clen} - \mathsf{macsize} - \mathsf{padlen}]_B$
    $\mathsf{out} \leftarrow \mathsf{out} \parallel (\top, m')$
    $\mathsf{d\text{-}seqnr} \leftarrow \mathsf{d\text{-}seqnr} + 1$
return out
```

Figure 8: The Simulator $\mathcal{S}$ used to prove Theorem 7.1.