Authenticated Encryption in Theory and in Practice

Jean Paul Degabriele

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics
Royal Holloway, University of London

2014

Declaration

These doctoral studies were conducted under the supervision of Prof. Kenneth G. Paterson.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Jean Paul Degabriele
August, 2014

Acknowledgements

First and foremost, I would like to thank my supervisor Kenny Paterson. It was with great luck that I met Kenny during my M.Sc. studies at Royal Holloway. He encouraged my interest in research and gave me the opportunity to pursue a Ph.D. I am deeply grateful for his guidance, support, and dedication throughout my studies, both on an academic and personal level.

A major part of the work presented in this thesis is the product of fruitful collaborations with Alexandra Boldyreva and Martijn Stam. I also had the chance to work with Anja Lehmann, Nigel Smart, Mario Strefler, and Gaven Watson. It was a great pleasure to work with you all. I am also indebted with Jonathan Katz who hosted me in his group at the University of Maryland.

I am glad to have pursued my studies at Royal Holloway; the Information Security Group has always provided me with a friendly and vibrant environment. I have made many friends here, but special thanks go to Anastasia, Charles, Eduarda, Gaven, Liz, Nadhem, Pooya, and Susan. I have also had some remarkable housemates with whom I shared many interesting conversations and who provided me with great support: Andrew, Ella, David, Giovanna, Jamie, Maithi, Peter, Prapti, Roe, and Vallu. Thank you all for making this experience so enjoyable.

Finally, I would like to thank my mother who always supported me through all my endeavours. I dedicate this to you.

Abstract

Authenticated encryption refers to a class of cryptographic schemes that simultaneously provide message confidentiality and message authenticity. It is an essential component of almost every cryptographic protocol that is used in practice. In this thesis we aim to narrow the gap that exists between authenticated encryption as used in practice, and authenticated encryption as studied in the framework of theoretical cryptography. We examine how certain types of attacks are not captured by the current techniques, and show how this can be remedied by expanding existing security models to capture a wider array of attacks.

We begin with a case study of IPsec: a widely deployed security protocol for protecting data across the Internet and other networks. Despite its popularity, IPsec's security has not received much formal treatment. As a security protocol it offers a relatively high degree of configurability, so as to accommodate multiple usage scenarios. We here present a new set of efficient attacks that fully break the confidentiality of half of the configurations that are permitted by the IPsec standard.

Next we turn our attention to the enhancement of security models. In particular we consider attacks that exploit distinguishable decryption failures and ciphertext fragmentation. A number of recent attacks against practical cryptosystems, including our attacks on IPsec, fall in one of these two categories. We extend the current security models to capture such attacks, and formulate new security notions to capture vulnerabilities that arise in this new setting. We then go on to explore how these notions relate to each other, and construct authenticated encryption schemes that satisfy our security notions.

Contents

Chapter 1

Introduction

Contents

This chapter gives an overview of the thesis. We provide the motivation for our research and describe the overall structure of this thesis.

1.1 The Evolution of Modern Cryptography

In recent years cryptography has flourished and it is increasingly present in various aspects of our everyday lives. Cryptography dates back to ancient times, when it mostly dealt with secret communication, but has now grown to span topics such as digital signatures, anonymity, zero knowledge proofs, secure computation, digital currency, digital rights management, and electronic voting. Nonetheless in today's world, centred on telecommunications, the need for confidentiality has never been as crucial. For hundreds of years cryptography was an underground black art, studied and practiced only by few. Cryptography attracted more interest with the advent of wireless communication and the Second World War, but it remained a taboo subject and was not studied freely within academia. This started to change towards the late 70's with the discovery of public-key cryptography [35], and has grown at a drastic rate ever since.

Once cryptography could be studied and used freely, it attracted the attention of mathematicians, computer scientists, and engineers. These studied the topic from different angles, and to some extent took it in different directions. To mathematicians cryptography offered a set of interesting problems which could be solved using number theory, combinatorics and coding theory, amongst others. Theoretical computer scientists, mainly complexity theorists, were more interested in fundamental questions such as the complexity theoretic assumptions that cryptographic applications could be based on, and which cryptographic protocols are realisable. Finally, engineers were interested in developing ‘real world’ systems, and were mainly driven by the need for such systems. However, engineers have often unintentionally sacrificed security for usability and practicality. While this is a far too simplistic generalisation, it serves to give an idea of some of the driving forces that shaped cryptography and which to some extent still exist to this day. In [87] Rogaway gives an account of his experience at MIT, describing how cryptography was seen as a branch of theoretical computer science that did not much countenance pragmatic concerns. To give an idea of the divide that existed between communities, he explains how after pursuing a PhD in cryptography at MIT he was totally unaware of Kerberos [76] which was developed by a team of engineers (outside of the theory group) at MIT.

The distinction between the engineers’ view of cryptography as opposed to the computer scientists’ view is quite clear and more or less translates to the divide between practitioners and theoreticians. The mathematician’s view lies somewhere in between and overlaps both ends of the spectrum. In particular, theoretical computer science is often regarded as a branch of mathematics, which makes this distinction less clear. Cryptography has undoubtedly benefited immensely from various branches of mathematics, but often mathematicians have viewed cryptography as an application of mathematics without much concern about the subject itself. It was complexity theorists who put the subject on more formal grounds, and established Modern Cryptography as we know it today. More specifically they developed the *Provable Security* approach which is considered to have transformed cryptography from an art to a science. The key ingredient to this approach is formal security definitions that permit a rigorous security analysis of cryptographic protocols. Provable security was formulated using ideas from complexity theory where an algorithm is considered to be efficient if it runs in polynomial time and security is proven using reductions. This rendered the provable security approach somewhat esoteric, and it was perceived by many to be detached from reality. Furthermore provable security was not concerned with symmetric key cryptography, which is essential to ‘real world' cryptography.

In the 90's, Bellare and Rogaway set to address this discrepancy in their research agenda. Their line of research, which they termed *practice-oriented provable security* [9, 87], applies the provable security approach to practical cryptographic schemes, and formulates its results in a way that is more meaningful to practice. Notable contributions of their approach include concrete security, the treatment of symmetric-key cryptography, and the random oracle methodology [19]. Their starting point was to abandon asymptotics and instead formulate security in a concrete manner. This simple alteration was pivotal in permitting the treatment of symmetric key cryptography, since block ciphers have no security parameter and hence it is not possible to define their security asymptotically. In addition, in the concrete setting security theorems are formulated quantitatively rather than qualitatively, allowing implementers to calculate key lengths and other parameter values for their required level of security. Another key contribution of their work was to model block ciphers as *pseudorandom permutations*. This has now become the accepted basic security requirement for block ciphers, and has replaced and extended Shannon's older formulation in terms of *confusion* and *diffusion* [89]. The random oracle methodology, while being a big source of controversy [26], has permitted a formal treatment of many practical schemes and has encouraged the exploration of security notions for hash functions that go beyond collision resistance and pre-image resistance. At a more philosophical level, entrenched in the practice-oriented provable security approach [87], is the view that cryptography is socially constructed as opposed to a view of cryptography conforming to scientific realism as the one expressed in [47] for example.

1.2 Motivation

Despite the efforts associated with practice-oriented provable security, provable security remains a source of controversy and divide within the cryptographic community. One such example is the long-standing controversy that originated from the paper by Koblitz and Menezes [67]. The authors sustain that the term 'provable' in provable security is misleading and gives a false sense of 100% certainty, concealing the conditional nature of proofs by reduction. In addition they point at cases where proofs were found to be flawed, reductions were not tight, or provably-secure schemes succumb to attacks outside the security model. In the conclusion to their paper, Koblitz and Menezes refer to these cases as ‘evidence’ that cryptographic protocol design is no more of a science than it is an art, thereby somehow discrediting provable security. In our view, Koblitz and Menezes raise many valid points but their conclusion is unfair. Reduction tightness and proof validity are evidenced because of the provable security approach in the first place, and thereby provide a better means for validating and contrasting the security of cryptographic schemes. The adequacy of security models [31, 68] on the other hand we believe to be a legitimate concern that needs to be addressed. The practical utility of provable security results is limited by the security models that they employ. Our view is that provable security has turned protocol design into a science, but crafting security definitions remains an art and security models need to evolve as new practical settings and attacks emerge.

In this thesis we will develop and study security models that are more relevant to practical settings. Our focus will be *authenticated encryption* schemes which combine message confidentiality and data origin authenticity. Authenticated encryption has become an essential component of almost every practical cryptographic protocol. In particular we will look at authenticated encryption as used in network protocols. We will first present novel attacks on IPsec that exploit features not normally captured in current security models. In particular the attacks will exploit *distinguishable decryption failures* in authenticated encryption schemes, and one attack will also exploit *ciphertext fragmentation*. Distinguishable decryption failures relate to an adversary’s ability to distinguish distinct decryption failure events. Such a vulnerability is usually inadvertently introduced by implementations and not normally taken into account in the theoretical analysis. Ciphertext fragmentation on the other hand, relates to the very nature of the interface presented by IP networks over which cryptographic protocols operate. In this setting, ciphertexts may be delivered at the receiver’s end in an arbitrarily fragmented fashion. It turns out that adapting authenticated encryption schemes to operate in such settings is not trivial, and this setting raises novel security concerns that are of interest in their own right. We will provide an extensive formal treatment of both distinguishable decryption failures and ciphertext fragmentation, by defining appropriate security notions, establishing relations and separations between security notions, and present constructions that meet these notions.

1.3 Thesis Structure

Chapter 2 starts with an overview of the provable security approach and the principles on which it is based. We then fix the notation and introduce some symmetric building blocks, namely block ciphers and message authentication schemes. This is followed by defining symmetric encryption and its security notions together with some basic block cipher modes of operation. We then look at authenticated encryption, and how to achieve it through generic composition. The chapter ends with extensions of security notions for symmetric encryption to the stateful setting.

In **Chapter 3** we lay down the practical background required for this thesis. We start off with a description of the TCP/IP protocol suite with a focus on some of its intricate aspects that are relevant to our work. We then go on to describe the three most popular secure network protocols that are in use today: TLS, SSH and IPsec. The focus of our exposition is primarily on the authenticated encryption component in each of these protocols. This is followed with a survey of previous works examining the security of these protocols. We conclude the chapter with a description of Vaudenay's padding oracle attack [93] and the SSH attack by Albrecht, Paterson, and Watson [1].

Chapter 4 presents a set of new attacks on various MAC-then-encrypt configurations of IPsec. We describe in detail three attack strategies, none of which is universally successful on all IPsec MAC-then-encrypt configurations, but for each configuration at least one strategy is successful. We also give an account of our experience in implementing the attacks on an experimental IPsec set-up.

In **Chapter 5** we provide a formal treatment of distinguishable decryption failures in symmetric encryption schemes. We revisit the classic relations for obtaining chosen-ciphertext security from chosen-plaintext security and integrity of ciphertexts in the light of distinguishable decryption failures. We establish a number of relations and separations between security notions in this setting. The chapter ends with a re-examination of the encrypt-then-MAC and MAC-then-encrypt compositions, providing further formal grounds for preferring the former composition.

Finally in **Chapter 6** we study symmetric encryption schemes supporting ciphertext fragmentation. We start off by defining the syntax for such schemes. This alone proves to be non-trivial. We then identify three notions of security, relating to confidentiality, boundary hiding, and denial of service. For each notion we consider a weaker variant that can be met by a subclass of schemes that maintain a 'minimal' state. The chapter concludes with constructions from standard symmetric primitives of schemes that meet our notions of security.

1.4 Associated Publications

Chapter 4 is joint work with Kenneth G. Paterson, and was published in the proceedings of the ACM CCS 2010 conference [30]. Chapters 5 and 6 are based on joint work with Alexandra Boldyreva, Kenneth G. Paterson, and Martijn Stam. Chapter 5 appeared in FSE 2013 [23], while the material of Chapter 6 was presented at Eurocrypt 2012 [22]. All authors conributed equally to the above publications.

Chapter 2

Symmetric Encryption

Contents

This chapter lays the required background on symmetric encryption. We present the basic schemes and define various notions of security.

2.1 Preliminaries

2.1.1 Provable Security

Provable security refers to a paradigm in cryptography which permits us to rigorously analyse and assess the security of cryptographic schemes and protocols. It is often claimed to have revolutionised cryptography from a ‘black art’ into a science. The approach borrows ideas from theoretical computer science and is based on two important concepts which form its heart and soul. The first is the formulation of *security definitions*. These specify the goals that cryptographic schemes are intended to achieve, such as confidentiality and data-origin authenticity, in precise mathematical terms. A security definition is normally expressed as an *experiment* played by an *adversary* with respect to a cryptographic scheme. Commonly the adversary is an algorithm parametrised by its computational resources. We are then concerned with the probability that an adversary has of 'winning' the experiment. The security of a scheme is expressed in terms of the maximal winning probability over some class of adversaries. Alternatively we say that a scheme is 'broken' if there exists a 'feasible' adversary whose winning probability exceeds the acceptible treshold by a significant amount.

The second pillar of provable security is the *reduction proof technique*. A proof of security generally boils down to bounding the winning probability of some class of adversaries. For most cryptographic schemes and protocols, proving such bounds without further assumptions appears to be currently beyond our reach. The difficulty in proving such bounds is related to the hardness of proving that $\mathcal{P} \neq \mathcal{NP}$. As a result security proofs are normally conditional, in that they rely on number-theoretic assumptions or on the security of the underlying primitives which the scheme is built on. This is where the reduction proof technique comes in. A reduction transforms an algorithm that solves problem B into an algorithm that solves problem A. We normally require that the transformation be efficiently computable. If such a transformation exists we say that problem A reduces to problem B. Now consider a scheme that is based on a block cipher. In order to prove its security we construct a reduction that transforms any adversary which breaks the scheme's security to an adversary that breaks the security of the block cipher. By assumption the block cipher is secure. Hence, by a contrapositive argument, it follows that the scheme is also secure. Put differently, the reduction shows that the only way to break the scheme is to find a flaw in the underlying primitive, in this case the block cipher. As a consequence of the reduction we need not bother with the cryptanalysis of the scheme, and we can focus instead on the cryptanalysis of the underlying primitive. Block ciphers like AES and DES have been thoroughly analysed for some years now and we are quite confident about their security. The reductionist approach allows us to transfer this confidence onto new cryptographic schemes that are built from them.

It is worth emphasising that such reductions are only possible because we have formal security definitions that permit a mathematical treatment. The design of security definitions is an intricate craft that is often underestimated and not given sufficient attention. Security definitions need to be simple enough to be mathematically manageable while at the same time encompass the lines of attack that are possible in complex practical settings. Every so often practice points out applications for which current security definitions turn out to be inadequate and require reconsideration. This 'gap' between theory and practice is a major theme of this thesis. In the following chapters we will see practical settings for which current security notions are inadequate and we will attempt to resolve this gap. A more in-depth discussion about the role and impact of definitions in the field of cryptography can be found in [86].

There are other paradigms in cryptography that fall under the umbrella of provable security. One such example is the information-theoretic approach, where the term 'provable' makes even more sense since such security proofs are unconditional. However the computational paradigm which we just described tends to be more relevant to practice and allows for certain types of cryptography, such as public key cryptography, that are simply not possible in the information-theoretic setting. The origin of the computational paradigm is normally attributed to the work of Goldwasser and Micali from 1982 [50]. Being a field mainly led by the theoretical community it initially evolved in a complexity-theoretic framework where algorithms are regarded as 'efficient' if they run in *polynomial time* and adversarial success probabilities are considered to be 'acceptable' if bounded by a *negligible* function. A security parameter is normally introduced such that running time and success probabilities can be expressed as functions of this parameter. Expressed in this framework, provable security results are not very meaningful to cryptographic practice. For instance, in symmetric cryptography block ciphers have fixed dimensions, and there is no security parameter that one can vary in order to obtain the required security level.

A more practically relevant approach was developed in a set of papers by Bellare and Rogaway [18, 15, 14]. As opposed to the asymptotic approach where a scheme is secure if all polynomial-time adversaries have a negligible success probability, their approach quantifies the success probability in terms of the adversary's resources. This is often referred to as *concrete security.* This approach entails a number of important differences. First and foremost the computational model that is assumed by the adversary becomes relevant in the concrete setting. Generally some RAM computational model is assumed as these better capture the architecture of modern computers. Security is now quantitative rather than just qualitative, and it allows us to better compare and contrast the security of cryptographic schemes. Another aspect that is surfaced by the concrete setting is the idea of reduction *tightness*. A reduction is essentially an algorithm, and the more efficient it is the tighter the reduction. In general for a specific scheme, tighter reductions yield better security bounds and are therefore desirable.

While the provable security paradigm is applicable to many sub-areas of cryptography, in this thesis we are mainly concerned with symmetric cryptography. We now introduce some notation, and then present the primitives and cryptographic schemes together with their corresponding security definitions that will be used in later chapters. Our treatment will mainly be in the concrete setting.

2.1.2 Notation

Unless otherwise stated, an algorithm may be randomised. An adversary is an algorithm. For any algorithm $\mathcal{A}$ we use $y \leftarrow \mathcal{A}(x_1, x_2, \dots)$ to denote executing $\mathcal{A}$ with fresh coins on inputs $x_1, x_2, \dots$ and assigning its output to y. If $\mathcal{S}$ is a set then $|\mathcal{S}|$ denotes its size, and $y \leftarrow_{\$} \mathcal{S}$ denotes the process of selecting an element from $\mathcal{S}$ uniformly at random and assigning it to y. The set of all finite binary strings is denoted by $\{0,1\}^*$. For any positive integer n and bit b, we denote by b^n the string of n consecutive b's and $\{0,1\}^n$ represents the set of all binary strings of length n. The empty string is represented by ε. For any two strings w and z and positive integers i and j, $w \parallel z$ denotes their concatenation, $w \oplus z$ denotes their bitwise XOR, $w \diamond z$ denotes the greatest common prefix of w and z, $w \,\%\, z$ denotes the remainder string of w with respect to $w \diamond z$ (i.e. $w = w \diamond z \parallel w \,\%\, z$), and $|w|$ denotes the length of w. Unless stated otherwise, $w[i]$ denotes the i^{th} bit of w, and $w[i,j]$ denotes the substring $w[i] \parallel w[i+1] \parallel \dots \parallel w[j]$. For any $n \in \mathbb{N}$, and any vector of strings $\mathbf{w} = [w_1, w_2, \dots, w_n]$, we define the concatenation orperator as $||(\mathbf{w}) = w_1 \parallel w_2 \parallel \dots \parallel w_n$. If j is a non-negative integer, then $\langle j \rangle_\ell$ denotes the unsigned ℓ-bit binary representation of j. Accordingly $\langle \cdot \rangle^{-1}$ represents the inverse mapping which maps strings of any length to $\mathbb{N}$. If w is an ℓ-bit string and i is an integer we use $w + i$ as shorthand for $\langle \langle w \rangle^{-1} + i \bmod 2^{\ell} \rangle_{\ell}$. We use $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ to denote the set of all functions with domain $\mathcal{X}$ and codomain $\mathcal{Y}$. Similarly $\mathsf{Perm}(\mathcal{X})$ denotes the set of all permutations over the domain $\mathcal{X}$. We will often have that $\mathcal{X} = \{0,1\}^{\ell}$ or $\mathcal{X} = \{0,1\}^{*}$, and $\mathcal{Y} = \{0,1\}^{n}$ for some positive integers ℓ and n. Accordingly we abbreviate notation for the corresponding sets of functions and permutations to $\mathsf{Func}(\ell, n)$, $\mathsf{Func}(*, n)$, and $\mathsf{Perm}(\ell)$ respectively. A list $\mathtt{L}$ is a triple $(\mathcal{I}, \mathcal{S}, f)$, where $\mathcal{I}$ is a set of integers acting as indexes, $\mathcal{S}$ is a set containing the members of the list, and f is a bijection mapping indexes to members of the list. We use () to denote the empty list, i.e. the list where $\mathcal{I} = \emptyset, \mathcal{S} = \emptyset$. For an integer i we denote by $\mathtt{L}_i$ the element of the list $f(i)$, and use $\mathtt{L}_i \leftarrow w$ to denote the process of assigning $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$ and $\mathcal{S} \leftarrow \mathcal{S} \cup \{w\}$, and letting $f(i) = w$. In addition we may apply set operators to the list, treating it as the set $\mathcal{S}$. Finally $\Pr[\, P : E \,]$ denotes the probability of event E occurring after having executed process P.

2.2 Building Blocks

2.2.1 Block Ciphers

Block ciphers are central to symmetric cryptography, and constitute an essential component of almost any cryptographic protocol that is used in practice. Their proliferation can probably be best attributed to the fact that they constitute a simple and yet versatile primitive with efficient implementations. Block-cipher design has been an active area of research for more than 30 years and is a relatively mature area of study within cryptography. In addition, popular block ciphers like DES and AES have been intensively cryptanlysed and to this day no severe security flaw has been discovered in their design. This serves as empirical evidence to show that block ciphers can be realised securely in practice and therefore constitute a good primitive on which to build other schemes. Unfortunately this is the best we have currently, since as was already alluded to in Section 2.1.1, proving that a block cipher is unconditionally secure would imply that $\mathcal{P} \neq \mathcal{NP}$.

Formally a block cipher is a function $E : \{0,1\}^{k} \times \{0,1\}^{n} \to \{0,1\}^{n}$ which takes a k-bit string and an n-bit string as input and returns an n-bit string as its output.

The *key length* k and *block length* n are parameters associated to the block cipher, and vary according to the block cipher's design. For each key $K \in \{0,1\}^k$ we let the function $E_K : \{0,1\}^n \to \{0,1\}^n$ be defined by $E_K(X) = E(K, X)$. Accordingly we can view a block cipher as a family of functions where each function is identified by the key. For any block cipher E, and any key K, it is generally required that the function E_K be a permutation, implying the existence of an inverse function E_K^{-1} such that $E_K^{-1}(E_K(X)) = X$. In terms of security what we normally require from block ciphers is for them to constitute good *pseudorandom functions* or good *pseudorandom permutations*. Pseudorandom functions were introduced by Goldreich, Goldwasser and Micali in [48, 49], but interestingly it was not until the work of Bellare, Kilian, and Rogaway that they were used to model block ciphers [15].

Definition 2.1: Pseudorandom Functions. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function family indexed by the set $\mathcal{K}$. Consider an adversary $\mathcal{A}$ with oracle access to some function with domain $\mathcal{X}$ and codomain $\mathcal{Y}$, and which returns a single bit as its output. We define the prf-advantage of adversary $\mathcal{A}$ with respect to the function family F as:

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) = \Pr\left[\, K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F_K(\cdot)} = 1 \,\right] - \Pr\left[\, f \xleftarrow{\$} \mathsf{Func}(\mathcal{X}, \mathcal{Y}) : \mathcal{A}^{f(\cdot)} = 1 \,\right].$$

F is said to be a pseudorandom function (PRF), if for every adversary $\mathcal{A}$ with reasonable resources its prf-advantage $\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A})$ is small.

Since this is the first security definition that we present, a few remarks are in order. The advantage represents the adversary's success probability; in this case its ability to distinguish between the two experiments. The advantage may be specified with respect to a single adversary or a class of adversaries. The latter allows us to quantify the security of a primitive or scheme, by the maximum advantage over all adversaries with resources bounded by some set of values $\mathcal{R}$. The resources of interest are usually the number of oracle queries that the adversary makes, the size of the queries, and the adversary's running time. By convention [15], the running time of the adversary includes its actual running time and the length of the RAM program that describes the adversary. This convention is intended to eliminate pathologies caused by adversaries which embed arbitrarily large lookup tables in their description. We will denote the maximum advantage by $\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{R})$, or simply

$\mathbf{Adv}_F^{\mathrm{prf}}$ to denote an absolute bound over all adversaries with reasonable resources. Security theorems will relate the advantage of a scheme to the advantages of its constituent primitives. It is then up to protocol designers to plug in advantage estimates for their primitives and interpret *small* and *reasonable* according to their application requirements.

Let us now turn our attention to the intuitive notions that the above security definition aims to capture. Note that picking a function at random from the set Func is equivalent to assigning $f(x)$ a uniformly random value from the set $\mathcal{Y}$ for each input $x \in \mathcal{X}$. The above definition states that a function family F is pseudorandom, if no computationally feasible adversary can distinguish effectively between a random instance of F and a random function. It then follows that, to an adversary, the PRF's output will appear random and uncorrelated both from the input and from other outputs. Thus pseudorandom functions capture the properties that we normally require from block ciphers. However a block cipher can be more accurately described as a collection of permutations; accordingly we can define pseudorandom permutations in a similar manner.

Definition 2.2: Pseudorandom Permutations. Let $\Pi : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a permutation family indexed by the set $\mathcal{K}$. Consider an adversary $\mathcal{A}$ with oracle access to some permutation over the set $\mathcal{X}$, and which returns a single bit as its output. We define the prp-advantage of adversary $\mathcal{A}$ with respect to the function family Π as:

$$\mathbf{Adv}_{\Pi}^{\mathrm{prp}}(\mathcal{A}) = \Pr\left[K \leftarrow_{\$} \mathcal{K} : \mathcal{A}^{\Pi_K(\cdot)} = 1\right] - \Pr\left[\pi \leftarrow_{\$} \mathsf{Perm}(\mathcal{X}) : \mathcal{A}^{\pi(\cdot)} = 1\right].$$

Π is said to be a pseudorandom permutation (PRP), if for every adversary $\mathcal{A}$ with reasonable resources its prp-advantage $\mathbf{Adv}_{\Pi}^{\mathrm{prp}}(\mathcal{A})$ is small.

Similarly to the previous case, we can think of a random permutation as assigning a random value to $\pi(x)$ for each $x \in \mathcal{X}$, with the sole distinction that the output values are now sampled from $\mathcal{X}$ and without replacement. The above notion can be strengthened by additionally providing the adversary with access to the inverse permutation (in each case). Permutation families which meet this stronger notion are called Strong Pseudorandom Permutations (SPRP). While it is more natural to model block ciphers as (S)PRPs, often their analysis may be considerably simpler if modelled as PRFs. The following lemma [15] addresses precisely this issue. In essence it says that for any good block cipher the prp-advantage and the prf-advantage are always close; i.e. they do not differ by more than the amount given by the birthday attack. Thus for sufficiently large block sizes, PRPs make good PRFs.

Result 2.1: PRP/PRF Switching Lemma cf. [15, Proposition 2.5].
Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a function family indexed by the set $\mathcal{K}$. Let $\mathcal{A}$ be an adversary that makes at most q queries to its oracle. Then

$$\left|\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) - \mathbf{Adv}_F^{\mathrm{prp}}(\mathcal{A})\right| \leq \frac{q(q-1)}{2|\mathcal{X}|}\,.$$

2.2.2 Message Authentication

Message authentication allows communicating parties who share a secret key to verify that a received message indeed originates from the party who claims to have sent it. Apart from providing a solution to this specific cryptographic problem, message authentication schemes also serve as a useful primitive for constructing other cryptographic protocols.

A *message authentication scheme* $\mathcal{MA} = (\mathcal{K}, \mathcal{T}, \mathcal{V})$ is a triple of algorithms with an associated message space $\mathcal{M} \subseteq \{0,1\}^*$. The randomised *key-generation* algorithm $\mathcal{K}$ takes no input and returns a secret key K. We will sometimes use $\mathcal{K}$ to denote the set of keys that may be output by the key-generation algorithm. The *tagging* algorithm $\mathcal{T}$ which may be randomised or stateful, takes as input a secret key $K \in \mathcal{K}$, and a message $m \in \mathcal{M}$, and returns a tag $\tau \in \{0,1\}^*$. The deterministic *verification* algorithm $\mathcal{V}$ takes as input the secret key $K \in \mathcal{K}$, a message $m \in \mathcal{M}$, and a candidate tag τ', to return a symbol $v \in \{\texttt{valid}, \perp\}$ denoting whether τ' is a valid tag for m or not. We require that for any key $K \in \mathcal{K}$ and any $m \in \mathcal{M}$

$$\Pr\left[\,\tau \leftarrow \mathcal{T}_K(m) : \mathcal{V}_K(m,\tau) = \texttt{valid}\,\right] = 1.$$

A number $\ell_{tag} \geq 1$ is called the *tag length* associated to the scheme if for any key

$K \in \mathcal{K}$ and any $m \in \mathcal{M}$

$$\Pr\left[\, \tau \leftarrow \mathcal{T}_K(m) : |\tau| = \ell_{tag} \,\right] = 1.$$

A special type of message authentication scheme which is by far the most common in practice, is the *message authentication code* (MAC). A MAC has the additional property that its tagging algorithm is stateless and deterministic, and its verification algorithm is given by:

```
$\underline{\mathcal{V}_K(m,\tau)}$
  $\tau' \leftarrow \mathcal{T}_K(m)$
  if $(\tau' = \tau)$ then return valid
  return $\perp$
```

The standard security notion for message authentication schemes is existential unforgeability under chosen message attacks (UF-CMA). This is an adaptation to the symmetric setting of the corresponding notion for signature schemes introduced by Golwasser, Micali, and Rivest [51]. An adversary is allowed to obtain tags for some number of messages of its choice, and wins if it can output a *new* message together with a valid tag. A stronger variant of this notion is termed strong unforgeabily under chosen message attacks (SUF-CMA). In this variant the adversary is also granted a 'win' if it can forge a new tag for a previously queried message. We define the two notions more formally below.

Definition 2.3: (S)UF-CMA. Let $\mathcal{MA} = (\mathcal{K}, \mathcal{T}, \mathcal{V})$ be a message authentication scheme. For an adversary $\mathcal{A}$, define experiments $\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A})$ and $\mathbf{Exp}^{\text{suf-cma}}_{\mathcal{MA}}(\mathcal{A})$ as shown in Figure 2.1. In both experiments, a key K is first generated by calling $\mathcal{K}$. The adversary $\mathcal{A}$ is then given access to a tagging oracle $\mathsf{Tag}(\cdot)$ and a verification oracle $\mathsf{Ver}(\cdot,\cdot)$. In $\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A})$ the adversary wins if it makes a successful verification query for some message which it had not previously queried to the tagging oracle. In $\mathbf{Exp}^{\text{suf-cma}}_{\mathcal{MA}}(\mathcal{A})$ the adversary wins if it makes a successful verification query for a message-tag pair that was not previously returned by the tagging oracle.

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

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

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

Figure 2.1: Experiments to define UF-CMA and SUF-CMA security. For UF-CMA the boxed code is excluded, whereas for SUF-CMA the boxed code replaces the code adjacent to it.

For each experiment we define the adversary's advantage as:

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

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

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

It is easy to see that SUF-CMA security implies UF-CMA security. Furthermore, any UF-CMA secure scheme may be amended by appending a redundant bit to its tags, and letting the verification algorithm ignore this bit. This simple alteration renders the scheme insecure in the SUF-CMA sense without detriment to its UF-CMA security, thereby yielding a (conditional) separation showing that SUF-CMA is a strictly stronger notion. However if we restrict ourselves to MACs, the two notions become equivalent. The separation we just sketched does not apply in the case of MACs. The verification procedure in the amended scheme does not conform to the MAC definition, and hence it does not qualify as a MAC. One further point to note about the above definitions is that we allowed the adversary to make multiple verification queries. In the cryptographic literature analogous notions can be found where the adversary is allowed only one verification query. It can be shown [13] that for the case of plain unforgeability, the multiple-query variant is strictly stronger, whereas for strong unforgeability the two variants are equivalent (up to a factor equal to the number of verification queries that the adversary is allowed). In recent work [36], Dodis, Kiltz, Pietrzak, and Wichs present a simple and efficient transformation yielding a UF-CMA secure message authentication scheme from any scheme that is unforgeable under a single verification query. Again, for the restricted case of MACs, security is not qualitatively affected by this variation in the definition.

We conclude this section on message authentication by pointing out a well-known connection with pseudorandom functions. A PRF with a sufficiently large codomain trivially yields a UF-CMA MAC, by letting the tagging algorithm be the PRF and defining the key-generation algorithm to sample a key for the PRF uniformly at random. This observation provides a pragmatic approach for realising MACs in practice; i.e. by extending readily-available PRFs (such as block ciphers) to accommodate the required domain. In fact many MAC designs like CMAC [91], XOR-MAC [14], and HMAC [10] can be viewed more simply as techniques for constructing variable-input-length PRFs from fixed-input-length PRFs.

2.3 Encryption Schemes

2.3.1 Syntax

A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of algorithms with an associated message space $\mathcal{M} \subseteq \{0,1\}^*$ and ciphertext space $\mathcal{C} \subseteq \{0,1\}^*$. The randomised *key-generation* algorithm $\mathcal{K}$ takes no input and returns a secret key K, an initial encryption state σ_0, and an initial decryption state ϱ_0. We will sometimes abuse notation and regard $\mathcal{K}$ as a set of keys representing the key space. The randomised and stateful *encryption* algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{M} \times \Sigma \to \mathcal{C} \times \Sigma$ takes as input the secret key $K \in \mathcal{K}$, a plaintext $m \in \mathcal{M}$, and the current encryption state $\sigma \in \Sigma$, and returns a ciphertext in $\mathcal{C}$ together with an updated state. The deterministic and stateful *decryption* algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{C} \times \Sigma \to (\mathcal{M} \cup \{\perp\}) \times \Sigma$ takes as input the secret key K, a ciphertext $c \in \mathcal{C}$, and the current decryption state ϱ to return the corresponding plaintext $m \in \mathcal{M}$ or the special symbol $\perp$ (indicating that the ciphertext is invalid) and an updated state.

For any $\ell \in \mathbb{N}$ and any $\mathbf{m} = [m_1, \ldots, m_\ell] \in \mathcal{M}^\ell$, we write $(\mathbf{c}, \sigma) \leftarrow \mathcal{E}_K(\mathbf{m}, \sigma_0)$ as shorthand for $(c_1, \sigma_1) \leftarrow \mathcal{E}_K(m_1, \sigma_0), (c_2, \sigma_2) \leftarrow \mathcal{E}_K(m_2, \sigma_1), \ldots (c_\ell, \sigma_\ell) \leftarrow \mathcal{E}_K(m_\ell, \sigma_{\ell-1})$, where $\mathbf{c} = [c_1, \ldots, c_\ell]$ and $\sigma = \sigma_\ell$. Similarly we use $(\mathbf{m}', \varrho) \leftarrow \mathcal{D}_K(\mathbf{c}, \varrho_0)$ to denote the analogous process for decryption.

For any symmetric encryption scheme we require that for all (K, σ_0, ϱ_0) that can be output by $\mathcal{K}$, all $\ell \in \mathbb{N}$, and all $\mathbf{m} \in \mathcal{M}^\ell$, it hold (with probability 1) that if $(\mathbf{c}, \sigma) \leftarrow \mathcal{E}_K(\mathbf{m}, \sigma_0)$ and $(\mathbf{m}', \varrho) \leftarrow \mathcal{D}_K(\mathbf{c}, \varrho_0)$, then $\mathbf{m}' = \mathbf{m}$.

The syntax for symmetric encryption that is presented here differs slightly from that which is commonly adopted in cryptographic literature. Our syntax defines both encryption and decryption to be stateful algorithms. Nonetheless, this does not result in any loss of generality. Both encryption and decryption can be made stateless by having $\mathcal{K}$ to always return the empty string for the corresponding initial state, and let the algorithm ignore (i.e. never update) the state. In later chapters we will consider practical aspects that will require further adjustments to the syntax of symmetric encryption schemes. In [85] Rogaway introduced a variation in the syntax in which encryption and decryption take a *nonce* as an additional input. While the nonce-based approach is certainly relevant to practice-oriented cryptography, it is somewhat orthogonal to the issues addressed in this thesis. For simplicity's sake we decided not to incorporate this approach in our analysis, yet it should be possible to extend most of our work to that setting.

2.3.2 Notions of Confidentiality

Bellare, Desai, Jokipii, and Rogaway were the first to consider formal notions of confidentiality for symmetric encryption [12]. They considered four candidate definitions, which they classified as *Find-then-Guess Indistinguishability*, *Left-or-Right Indistinguishability*, *Real-or-Random Indistinguishability*, and *Semantic Security*. Each notion is defined through an experiment. Find-then-Guess Indistinguishability is a straightforward adaptation of the well-known IND-CPA notion from the public-key setting, first introduced under the name 'polynomial security' in [50]. Here a two-stage experiment is defined, where in the first stage – the 'find' stage the adversary endeavours to come up with a pair of equal-length messages whose encryptions it wants to try to tell apart. In the 'guess' stage a message from this pair is chosen at random, and its encryption called the challenge ciphertext is given to the adversary. The adversary wins if it can tell which message is concealed in the challenge ciphertext. An encryption scheme is deemed secure if no adversary consuming reasonable resources can win significantly more than half the time.

Left-or-right indistinguishability is defined through a single-stage experiment which starts by sampling a bit uniformly at random. The adversary is then given access to a left-or-right oracle which, when queried on a pair of equal-length messages, returns the encryption of one of them. The message to be encrypted is chosen according to the bit sampled at the beginning of the experiment. The experiment ends when the adversary outputs a guess of the bit. Similarly as before, an encryption scheme is deemed secure if no adversary consuming reasonable resources can guess the bit significantly more than half the time. Real-or-random indistinguishability is defined similarly, except that the adversary is instead given access to a real-or-random oracle, which takes as input a single message, and according to the value of the sampled bit, either returns an encryption of the input or an encryption of an equal-length message sampled uniformly at random. Semantic security is also an adaptation to the symmetric setting of the notion introduced in [50] under the same name. All of these notions can be strengthened to capture chosen-ciphertext attacks by additionally giving the adversary access to a decryption oracle and prohibiting him from making 'trivial-win' queries.

The authors of [12] also establish relations among these four security notions. They show that left-or-right and real-or-random indistinguishability are equivalent up to a factor of two in the corresponding advantages. Similarly find-then-guess indistinguishability and semantic security are shown to be equivalent up to a constant factor of two. Finally they show that find-then-guess and left-or-right indistinguishability are also equivalent, but the reduction from left-or-right indistinguishability to find-then-guess indistinguishability incurs a loss in the advantage by a factor equal to the number of queries made by the adversary to the left-or-right oracle. Furthermore they present a separation showing that this reduction is tight, and hence this 'loss in security' is inevitable. The above relations apply equally to the chosen-ciphertext-attack variants of the notions. From the above analysis, left-or-right and real-or-random indistinguishability emerge as the preferred security notions, since they are quantitatively stronger. We now present in more detail definitions for $\mathsf{IND\text{-}CPA}$ and $\mathsf{IND\text{-}CCA}$ security in terms of left-or-right indistinguishability.

Definition 2.4: IND-CPA and IND-CCA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$ and a bit b, define experiments $\mathbf{Exp}^{\text{ind-cpa-b}}_{\mathcal{SE}}(\mathcal{A})$ and $\mathbf{Exp}^{\text{ind-cca-b}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 2.2. Both experiments start by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to a left-or-right encryption oracle $\mathsf{LoR}(\cdot)$, and in the latter experiment it is additionally given a decryption oracle $\mathsf{Dec}(\cdot)$. No restriction is imposed on the adversary's queries, rather if it queries a pair of messages of unequal length to $\mathsf{LoR}(\cdot)$, or if it queries a ciphertext to $\mathsf{Dec}(\cdot)$ previously returned by $\mathsf{LoR}(\cdot)$, the $\lightning$ symbol is returned.

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

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

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

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

Arguably, out of the four formulations of $\mathsf{IND\text{-}CPA}$ and $\mathsf{IND\text{-}CCA}$ security, semantic

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

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

$\underline{\mathsf{Dec}(c)}$
  $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
  $\mathbf{if}\ c \in \mathtt{C}\ \mathbf{then}\ m \leftarrow \lightning$
  $\mathbf{return}\ m$
```

Figure 2.2: Experiments to define IND-CPA and IND-CCA security. For IND-CPA the boxed code is excluded, whereas for IND-CCA the boxed code replaces the code adjacent to it.

security is intuitively the most satisfactory notion of confidentiality. It conveys the most clear intuition that for an encryption scheme to be secure, it should not leak any partial information about the plaintext. This intuition is not immediately clear from the other formulations. On the other hand, the other formulations tend to be easier to use in security proofs. This further highlights the utility of the equivalence relations from [12], in that they allow us to get the best of both worlds. Furthermore, the fact that different notions capturing seemingly different intuitive notions of confidentiality turn out to be equivalent is often considered a good indication that we have converged to a good security notion.

Symmetric encryption schemes that satisfy left-or-right indistinguishability (IND), often satisfy a stronger notion of indistinguishability known as *indistinguishability from random bits* (IND\$). This notion was put forward by Rogaway in [85]. Under this notion a scheme is secure if ciphertexts cannot be distinguished from random strings of the same length. Interestingly, for typical schemes, proving IND\$ security often turns out to be slightly simpler than proving IND security. Note however that $\mathsf{IND\$}$ implies IND only as long as the distribution of ciphertext lengths (for the scheme under consideration) depends only on the message length (and not the message itself). Analogous definitions for chosen-plaintext and chosen-ciphertext security are defined as follows:

Definition 2.5: IND\$-CPA and IND\$-CCA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$ and a bit b, define experiments $\mathbf{Exp}^{\text{ind\$-cpa-b}}_{\mathcal{SE}}(\mathcal{A})$ and $\mathbf{Exp}^{\text{ind\$-cca-b}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 2.3. Both experiments start by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to a special encryption oracle $\mathsf{Enc\$}(\cdot)$. If $b = 1$ the oracle returns the encrypted message, otherwise it returns a uniformly-random bit-string of the same length as the encrypted message. In the latter experiment it is additionally given access to a decryption oracle $\mathsf{Dec}(\cdot)$. Trivial-win conditions are avoided by having the decryption oracle return $\lightning$ in response to any ciphertext that was previously output by the encryption oracle.

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

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

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

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

2.3.3 Modes of Operation

We now describe three ubiquitous symmetric encryption schemes that are based on block ciphers. These are also known as *modes of operation*, since they are effectively

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

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

$\underline{\mathsf{Dec}(c)}$
  $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
  if $c \in \mathtt{C}$ then $m \leftarrow \lightning$
  return $m$
```

Figure 2.3: Experiments to define IND\$-CPA and IND\$-CCA security. For IND\$-CPA the boxed code is excluded, whereas for IND\$-CCA the boxed code replaces the code adjacent to it.

ways of operating a block cipher for encrypting data. We assume, in all three cases, that the message space contains only messages whose length is an integer multiple of the block length. The simplest mode of operation is the *Electronic Code Book* (ECB) mode, in which a message is split into blocks, each block is then fed into the block cipher, and the outputs are concatenated. Note that this mode of operation is deterministic, and consequently it cannot be IND-CPA secure irrespective of which block cipher is used.

A more secure mode of operation is *Counter* mode. In this mode, the input to the block cipher is a counter that is incremented after each block cipher call. The outputs of the block cipher are then concatenated and the resulting string is XORed to the plaintext in a fashion similar to the one-time pad. We here present two variants of this mode, of which one is stateful and the other randomised. For any block cipher E we denote *Stateful Counter* mode by $\mathsf{CTR}[E]$, and *Randomised Counter* mode by $\mathsf{CTR\$}[E]$. Namely in the stateful variant the counter is maintained as the encryption state, whereas in the randomised variant a random initial counter is

```
$\underline{\mathsf{CTR}\text{-}\mathcal{K}}$
    $K \leftarrow_{\$} \{0,1\}^k$
    **return** $(K, 0, \varepsilon)$

$\underline{\mathsf{CTR}\text{-}\mathcal{E}_K(m, \sigma)}$
    $c_0 \leftarrow \sigma$
    **for** $i = 1$ **to** $|m|/n$
        $x_i \leftarrow m[1 + (i-1)n, in]$
        $c_i \leftarrow E_K(c_0 + i) \oplus x_i$
    $c \leftarrow c_0 \parallel \ldots \parallel c_{|m|/n}$
    **return** $(c, \sigma + i)$

$\underline{\mathsf{CTR}\text{-}\mathcal{D}_K(c, \varrho)}$
    $y_0 \leftarrow c[1, n]$
    **for** $i = 1$ **to** $|c|/n - 1$
        $y_i \leftarrow c[1 + in, (i+1)n]$
        $m_i \leftarrow y_i \oplus E_K(y_0 + i)$
    $m \leftarrow m_1 \parallel \ldots \parallel m_{|c|/n-1}$
    **return** $(m, \varrho)$
```

Figure 2.4: Scheme $\mathsf{CTR}[E]$ from a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$.

```
$\underline{\mathsf{CTR\$}\text{-}\mathcal{K}}$
    $K \leftarrow_{\$} \{0,1\}^k$
    **return** $(K, \varepsilon, \varepsilon)$

$\underline{\mathsf{CTR\$}\text{-}\mathcal{E}_K(m, \sigma)}$
    $c_0 \leftarrow_{\$} \{0,1\}^n$
    **for** $i = 1$ **to** $|m|/n$
        $x_i \leftarrow m[1 + (i-1)n, in]$
        $c_i \leftarrow E_K(c_0 + i) \oplus x_i$
    $c \leftarrow c_0 \parallel \ldots \parallel c_{|m|/n}$
    **return** $(c, \sigma)$

$\underline{\mathsf{CTR\$}\text{-}\mathcal{D}_K(c, \varrho)}$
    $y_0 \leftarrow c[1, n]$
    **for** $i = 1$ **to** $|c|/n - 1$
        $y_i \leftarrow c[1 + in, (i+1)n]$
        $m_i \leftarrow y_i \oplus E_K(y_0 + i)$
    $m \leftarrow m_1 \parallel \ldots \parallel m_{|c|/n-1}$
    **return** $(m, \varrho)$
```

Figure 2.5: Scheme $\mathsf{CTR\$}[E]$ from a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$.

sampled uniformly for each ciphertext. In either case the initial counter is prepended to the ciphertext. The two modes are specified in Figures 2.4 and 2.5.

A third variant could be defined in which decryption is also stateful. Here the decryption algorithm maintains its own copy of the counter, and no initial counter value is prepended to the ciphertext. Note that all three variants can be easily adapted to support messages of arbitrary length. This is achieved by trimming the redundant part from the last block cipher output, such that the concatenated outputs are of the same length as the message.

The third mode of operation that we consider is *Cipher Block Chaining* (CBC) mode. In this scheme encryption is stateless and randomised, and decryption is also stateless. In CBC mode each plaintext block is first XORed with the previous ciphertext block, and the resulting block is then fed into the block cipher to yield the corresponding ciphertext block. The first ciphertext block is computed by XORing the plaintext block with an initial vector, sampled uniformly at random, and the result is then fed into the block cipher. This initial vector is prepended to the ciphertext so that the receiver can decrypt successfully. Decryption is the reverse process, each ciphertext block is inverted using the inverse block cipher function and the result is XORed to the preceding ciphertext block (or initial vector) to yield the corresponding plaintext block. In order to encrypt messages of arbitrary length, a padding scheme is normally used to extend the message length to an integer multiple of the block length. The $\mathsf{CBC}[E]$ scheme obtained from a block cipher E is specified in Figure 2.6.

The Counter mode variants and CBC mode were proven secure in [12]. Namely $\mathsf{CTR}[E]$ and $\mathsf{CTR\$}[E]$ were proven to be $\mathsf{IND\text{-}CPA}$ secure under the assumption that E is a pseudorandom function, while $\mathsf{CBC}[E]$ was shown to be $\mathsf{IND\text{-}CPA}$ secure under the assumption that E is a pseudorandom permutation. All three schemes suffer from some form of malleability, and as a consequence they cannot meet the stronger $\mathsf{IND\text{-}CCA}$ security notion. For both Counter mode variants, flipping a bit in the ciphertext results in the corresponding plaintext bit being flipped. Similarly in CBC mode, flipping a bit in a ciphertext block results in the corresponding bit of the next plaintext block being flipped. In addition if the block cipher is a PRP, we can expect that the current plaintext block will be randomised if any of the corresponding ciphertext-block bits are flipped. Thus an adversary can predictably alter the contents of a plaintext block at the expense of 'garbling' the preceding plaintext block. However the first block can be manipulated by flipping bits in the initial vector without garbling any other plaintext block. This malleability can be exploited in the $\mathsf{IND\text{-}CCA}$ experiment as follows. The adversary submits a message pair $(0^n, 1^n)$ to the left-or-right oracle and receives in return a ciphertext c. Using the techniques just described, it produces a ciphertext c' that corresponds to m with its first bit flipped, where m is the message contained in c. It then submits c' to the decryption oracle and if the returned message is $1 \,\|\, 0^{n-1}$ it outputs 0, else if it is $0 \,\|\, 1^{n-1}$ it outputs 1. Thereby the adversary wins the $\mathsf{IND\text{-}CCA}$ experiment with probability 1. In the coming chapters we will make use of this malleability to mount plaintext-recovery attacks on practical cryptosystems.

```
CBC-$\mathcal{K}$
    $K \leftarrow_{\$} \{0,1\}^k$
    return $(K, \varepsilon, \varepsilon)$

CBC-$\mathcal{E}_K(m, \sigma)$
    $c_0 \leftarrow_{\$} \{0,1\}^n$
    for $i = 1$ to $|m|/n$
        $x_i \leftarrow m[1 + (i-1)n, in]$
        $c_i \leftarrow E_K(c_{i-1} \oplus x_i)$
    $c \leftarrow c_0 \parallel \ldots \parallel c_{|m|/n}$
    return $(c, \sigma)$

CBC-$\mathcal{D}_K(c, \varrho)$
    $y_0 \leftarrow c[1, n]$
    for $i = 1$ to $|c|/n - 1$
        $y_i \leftarrow c[1 + in, (i+1)n]$
        $m_i \leftarrow y_{i-1} \oplus E_K^{-1}(y_i)$
    $m \leftarrow m_1 \parallel \ldots \parallel m_{|c|/n-1}$
    return $(m, \varrho)$
```

Figure 2.6: Scheme $\mathsf{CBC}[E]$ from a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$.

2.4 Authenticated Encryption

Until now we looked at message authenticity and message confidentiality as separate goals. In practice however, when two parties communicate over a network, both goals are desirable. Authenticated encryption refers to schemes which simultaneously achieve both message confidentiality and message authenticity. Historically the lack of authenticated encryption has been a source of problems for practical cryptosystems. A number of proposed schemes claimed to meet this goal have resulted in disastrous failures. For instance in early drafts of the IPsec protocol suite [4] it was claimed that schemes such as CBC mode provide both confidentiality and integrity. This misconception was in part due to a lack of formal security definitions, where message integrity and message authenticity (in the symmetric setting) were perceived to be different [84]. Another popular example is that of the WEP cryptosystem in the IEEE 802.11 standard, where a Cyclic Redundancy Checksum was combined with a stream cipher in an attempt to provide authenticated encryption. Both cases were later shown to be inadequate for message authenticity as well as message privacy [20, 29, 25]. These two examples (together with many others) raise a number of issues that are worth pointing out. Firstly, they highlight the importance of the provable security approach to cryptographic practice, as opposed to claims of security (as in [4]) based solely on intuition. Secondly they show that, at least at the time, it was not at all obvious how to construct authenticated encryptions schemes. Thirdly they also show that practical cryptographic applications often necessitate stronger confidentiality than that guaranteed by IND-CPA security.

Bellare and Namprempre [17] were amongst the first to provide a formal analysis of authenticated encryption. We now summarise some of their results. An authenticated encryption scheme is syntactically equivalent to a symmetric encryption scheme; its characterisation derives instead from the additional property that it simultaneously meets some appropriate notions of message confidentiality and message authenticity. Bellare and Namprempre identify IND-CCA security as the appropriate confidentiality notion, and define message authenticity in terms of the *integrity of plaintexts* (INT-PTXT) notion. They also consider a closely-related notion called *integrity of ciphertexts* (INT-CTXT). We now define both.

Definition 2.6: INT-PTXT and INT-CTXT. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$ define the experiments $\mathbf{Exp}^{\text{int-ptxt}}_{\mathcal{SE}}(\mathcal{A})$ and $\mathbf{Exp}^{\text{int-ctxt}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 2.7. Both experiments start by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$, and a try oracle $\mathsf{Try}(\cdot)$. When queried on a ciphertext, the try oracle returns whether that ciphertext is valid or not.

In the INT-PTXT experiment the adversary wins if it submits to the try oracle a valid ciphertext that does not correspond to a plaintext previously queried to the encryption oracle. In the INT-CTXT experiment however, the adversary wins if it submits to the try oracle a valid ciphertext not previously returned by the encryption oracle. For each experiment we define the advantage of an adversary $\mathcal{A}$ as:

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

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

The scheme $\mathcal{SE}$ is said to be INT-PTXT (or INT-CTXT) secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{int-ptxt}}_{\mathcal{SE}}(\mathcal{A})$ (respectively $\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}}(\mathcal{A})$) is small.

It is easy to see that for any symmetric encryption scheme, if it satisfies integrity of ciphertexts it will also satisfy integrity of plaintexts. More simply we say that integrity of ciphertexts implies integrity of plaintexts, which we denote by INT-CTXT $\longrightarrow$ INT-PTXT. Integrity of plaintexts is perhaps the more natural security requirement, but integrity of ciphertexts is interesting, at least in part, due to the following

```
$\underline{\mathbf{Exp}^{\text{int-ptxt}}_{\mathcal{SE}}(\mathcal{A})}$ $\boxed{\underline{\mathbf{Exp}^{\text{int-ctxt}}_{\mathcal{SE}}(\mathcal{A})}}$
  $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
  $\mathsf{win} \leftarrow 0, \mathsf{L} \leftarrow \emptyset$
  $\mathcal{A}^{\mathsf{Enc}(\cdot),\mathsf{Try}(\cdot)}$
  $\mathbf{return}\ \mathsf{win}$

$\underline{\mathsf{Enc}(m)}$
  $(c, \sigma) \leftarrow \mathcal{E}_K(m, \sigma)$
  $\mathsf{L} \leftarrow \mathsf{L} \cup m$ $\boxed{\mathsf{L} \leftarrow \mathsf{L} \cup c}$
  $\mathbf{return}\ c$

$\underline{\mathsf{Try}(c)}$
  $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
  $\mathbf{if}\ m \notin \mathsf{L}\ \mathbf{and}\ m \neq \perp$ $\boxed{\mathbf{if}\ c \notin \mathsf{L}\ \mathbf{and}\ m \neq \perp}$
    $\mathbf{then}\ \mathsf{win} \leftarrow 1$
  $\mathbf{if}\ m \neq \perp\ \mathbf{then}\ m \leftarrow \lightning$
  $\mathbf{return}\ m$
```

Figure 2.7: Experiments to define INT-PTXT and INT-CTXT security. For INT-PTXT the boxed code is excluded, whereas for INT-CTXT the boxed code replaces the code adjacent to it.

result from [17].

Theorem 2.2: IND-CPA $\wedge$ INT-CTXT $\longrightarrow$ IND-CCA. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For any* IND-CCA *adversary $\mathcal{A}_{cca}$ there exist adversaries $\mathcal{A}_{cpa}$ and $\mathcal{A}_{int}$, comsuming similar resources to $\mathcal{A}_{cca}$, such that:*

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(\mathcal{A}_{cca}) \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) + 2 \cdot \mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}}(\mathcal{A}_{int})\,. \tag{2.1}$$

Informally, the above theorem states that *weak* confidentiality (IND-CPA) together with ciphertext integrity, yields *strong* confidentiality (IND-CCA). Thus in order to obtain an authenticated encryption scheme it suffices to construct a scheme that is both IND-CPA secure and INT-CTXT secure. Bellare and Namprempre go on to show, by means of a separation, that an analogous implication where ciphertext integrity is replaced with plaintext integrity does not hold. They also show that IND-CCA does not guarantee INT-PTXT, and hence the converse of Theorem 2.2

does not hold either.

We have not yet given any example of an authenticated encryption scheme. A natural attempt at constructing one would be to somehow combine a message authentication scheme with a symmetric encryption scheme. This is known as *generic composition*. Three natural ways of combining the two primitives emerge, and as we shall see in the next chapter, all three have been deployed in practical cryptosystems. These are: Authenticate-then-Encrypt (AtE), Encrypt-then-Authenticate (EtA), and Encrypt-and-Authenticate (E&A). They are informally defined as follows:

Authenticate-then-Encrypt (AtE): The sender computes a tag on the plaintext, the tag is appended to the plaintext, and the resulting string is then encrypted. The receiver decrypts the ciphertext, recovers the tag and the plaintext, and if the tag verifies correctly it returns the plaintext, otherwise it returns $\perp$.

Encrypt-then-Authenticate (EtA): The sender encrypts the plaintext, computes a tag on the ciphertext, and appends the tag to the ciphertext. The receiver recovers the tag and the ciphertext. If the tag verifies correctly, the receiver goes on to decrypt the ciphertext and returns the plaintext, otherwise it returns $\perp$.

Encrypt-and-Authenticate (E&A): The sender computes a tag on the plaintext. It then encrypts the plaintext, and appends the tag to the ciphertext. The receiver recovers the tag and the ciphertext, and decrypts the ciphertext. If the tag on the resulting plaintext verifies correctly the plaintext is returned, otherwise it returns $\perp$.

Bellare and Nampremrpre [17] analysed these three compositions and showed that, for any IND-CPA secure encryption scheme and any SUF-CMA message authentication scheme, only encrypt-then-authenticate guarantees that the resulting scheme will be IND-CCA secure. Moreover, if the message authentication scheme is SUF-CMA secure it follows rather trivially that the EtA composition is INT-CTXT secure. Thus for any two primitives that satisfy these basic security requirements, EtA always yields an authenticated encryption scheme. Of course this does not mean that the other two compositions can never yield an authenticated encryption scheme; rather it says that the security mean that the other two compositions can never yield an authenticated encryption scheme; rather it says that the security of these compositions does not immediately follow from the IND-CPA and SUF-CMA security of the underlying primitives. In the case of encrypt-and-authenticate, the message authentication scheme may leak information about the plaintext and yet be SUF-CMA secure. Hence the composed scheme may not even be IND-CPA secure. As regards authenticate-then-encrypt, the encryption scheme may be susceptible to some form of malleability which allows an attacker to maul a ciphertext without altering the underlying plaintext and tag. Consequently the resulting composition cannot be IND-CCA secure. This intuitive argument can be formalised to yield a counterexample showing that AtE (and equally to E&A) does not guarantee IND-CCA security in general, as well as to prove the aforementioned separation, namely IND-CPA $\wedge$ INT-PTXT$\not\rightarrow$ IND-CCA.

Following the work of Bellare and Namprempre, a fair amount of research has been dedicated to various aspects of authenticated encryption. In particular, there has been an effort to construct dedicated schemes whose performance surpasses that of generic composition schemes. Examples of dedicated schemes are Integrity-Aware CBC (IACBC) [59], Counter mode with CBC-MAC (CCM) [97], Galois Counter Mode (GCM) [75], and Offset CodeBook (OCB) [88]. The design specifics of such dedicated schemes are beyond the scope of this thesis and we do not discuss them any further.

We conclude this section by presenting a security notion, put forward by Shrimpton [90], that elegantly combines IND-CPA and INT-CTXT into a single notion. This notion is known as IND-CCA3 or simply as AE security.

Definition 2.7: IND-CCA3. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$, and a bit b, define the experiment $\mathbf{Exp}^{\text{ind-cca3-b}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 2.8. First $\mathcal{K}$ is called to generate a key K, an initial encryption state σ, and an initial decryption state ϱ. The adversary $\mathcal{A}$ is then given access to a special encryption oracle $\mathsf{EncR}(\cdot)$ and a special decryption oracle $\mathsf{Dec}\varnothing(\cdot)$. When $b = 1$ both oracles behave as normal encryption and decryption oracles. Contrarily, if $b = 0$ then $\mathsf{EncR}(\cdot)$ will return the encryption of a random string of the same length

```
$\underline{\mathbf{Exp}^{\text{ind-cca3-b}}_{\mathcal{SE}}(\mathcal{A})}$
  $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
  $i \leftarrow 0,\ \mathtt{C} \leftarrow ()$
  $b' \leftarrow \mathcal{A}^{\mathsf{EncR}(\cdot),\mathsf{Dec}\varnothing(\cdot)}$
  return $(b')$

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

$\underline{\mathsf{Dec}\varnothing(c)}$
  $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
  if $b = 0$ then $m \leftarrow \perp$
  if $c \in \mathtt{C}$ then $m \leftarrow ⚡$
  return $m$
```

Figure 2.8: Experiment to define IND-CCA3 security.

as the message, and $\mathsf{Dec}\varnothing(\cdot)$ will always return $\perp$ (unless the queried ciphertext was output by $\mathsf{EncR}(\cdot)$, in which case it will return ⚡).

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

$$\mathbf{Adv}^{\text{ind-cca3}}_{\mathcal{SE}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{ind-cca3-1}}_{\mathcal{SE}}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca3-0}}_{\mathcal{SE}}(\mathcal{A}) = 1\right].$$

The scheme $\mathcal{SE}$ is said to be IND-CCA3 secure, if for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE}}(\mathcal{A})$ is small.

For a proof of the equivalence between IND-CCA3 and IND-CPA $\wedge$ INT-CTXT the reader is referred to [90]. Note that the IND-CCA3 notion is formulated using real-or-random indistinguishability. An analogous IND\$-CCA3 notion which combines IND\$-CPA and INT-CTXT, can be similarly defined by replacing the $\mathsf{EncR}(\cdot)$ oracle with the $\mathsf{Enc\$}(\cdot)$ oracle from Definition 2.5.

2.5 Stateful Security

A main security concern in practical cryptosystems is that of *replay* and *re-ordering* attacks. These are attacks characterised by an adversary's ability to replay and to re-order ciphertexts during their transmission. Consider for instance an online banking application where a message may constitute a payment instruction. By replaying the ciphertext that corresponds to a payment instruction, an adversary may be able to force the payment to occur multiple times. Alternatively consider the case where for transmission purposes messages need to be split into smaller message segments, which are then encrypted separately. By reordering the corresponding ciphertexts an adversary may be able to forge new reconstructed messages at the receiver side. These examples obviously constitute a breach of integrity, yet neither constitutes an attack under the notions of integrity that we have presented so far.

The ability to replay and re-order ciphertext can also impact confidentiality. Consider again the last example where an application transmits application messages in smaller segments over a network, and each message segment is encrypted separately. Further assume that the application leaks (by means of some side-channel) to the adversary a small amount of information about the received message. Now by re-ordering and replaying ciphertexts, an adversary may be able to inject new application messages that bear some relation to the original application message, and thereby amplify the amount of information leaked through the side-channel. Of course the cryptosystem cannot prevent the upper-layer application from leaking information to the adversary. However it can ensure for instance that replayed and re-ordered ciphertexts do not decrypt successfully and hence prevent further leakage of information from adversarially manipulated traffic. Note that IND-CCA security does not capture these security goals, and consequently it does not guarantee security against replay and re-ordering attacks.

As a tool in their analysis of SSH [16], Bellare, Kohno, and Namprempre extended chosen-ciphertext security and integrity of ciphertexts to additionally protect against replay and re-ordering of ciphertext. Their security notions, IND-sfCCA and INT-sfCTXT, can be seen as 'strengthened' variations of their standard counterparts. That is IND-sfCCA $\longrightarrow$ IND-CCA and INT-sfCTXT $\longrightarrow$ INT-CTXT. These definitions consider whether the adversary's decryption queries are *in-sync* with its encryption queries. Namely, as long as the sequence of ciphertexts queried for decryption (or to the Try oracle) is a prefix of the ciphertext sequence returned by the encryption (or left-or-right) oracle, the decryption queries are said to be in-sync. Alternatively as soon as a decryption query is made such that this relation no longer holds, the decryption queries are said to have become *out-of-sync*. Intuitively this can interpreted to say that as long as the decryption queries are in-sync the adversary is acting passively. Conversely if the decryption queries have become out-of-sync, it means that the adversary has tampered with the traffic flow between the sender and the receiver. IND-sfCCA is then defined analogously to IND-CCA except that the output of the decryption algorithm is only returned once the decryption queries have become out-of-sync. Note that this is less restrictive on the adversary than the IND-CCA restriction that ciphertexts returned by the left-or-right oracle cannot be queried to the decryption oracle so IND-sfCCA is a stronger security notion. Similarly in INT-sfCTXT any out-of-sync ciphertext that decrypts correctly is considered a ciphertext forgery. Thus, and in contrast to INT-CTXT, a replay of a ciphertext previously output by the encryption oracle can result in a 'win' for the adversary. We now define more formally the two notions.

Definition 2.8: INT-sfCTXT. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$ define the experiment $\mathbf{Exp}^{\text{int-sfctxt}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 2.9. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. An adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$, and a stateful try oracle $\mathsf{sfTry}(\cdot)$. When queried on a ciphertext, the try oracle returns $\lightning$ if the queried ciphertext is valid or it is in-sync, and returns $\perp$ if it is invalid. The adversary's goal is to make a valid out-of-sync try query, and its advantage is defined as:

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

The scheme $\mathcal{SE}$ is said to be INT-sfCTXT secure, if for every adversary $\mathcal{A}$ consuming reasonable resources, its advantage $\mathbf{Adv}^{\text{int-sfctxt}}_{\mathcal{SE}}(\mathcal{A})$ is small.

Definition 2.9: IND-sfCCA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$ and a bit b, define the experiment $\mathbf{Exp}^{\text{ind-sfcca-b}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 2.9. The experiment starts by calling $\mathcal{K}$ to generate a key K

```
$\mathbf{Exp}^{\text{int-sfctxt}}_{\mathcal{SE}}(\mathcal{A})$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, j \leftarrow 0$
    $\mathsf{C} \leftarrow (), \mathsf{sync} \leftarrow 1, \mathsf{win} \leftarrow 0$
    $\mathcal{A}^{\mathsf{Enc}(\cdot), \mathsf{sfTry}(\cdot)}$
    return $\mathsf{win}$

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

$\mathsf{sfTry}(c)$
    $j \leftarrow j + 1, (m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
    if $j > i$ or $c \neq \mathsf{C}_j$
        then $\mathsf{sync} \leftarrow 0$
    if $\mathsf{sync} = 0$ and $m \neq \perp$
        then $\mathsf{win} \leftarrow 1$
    if $m \neq \perp$ then $m \leftarrow \lightning$
    return $m$
```

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

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

$\mathsf{sfDec}(c)$
    $j \leftarrow j + 1$, $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
    if $j > i$ or $c \neq \mathsf{C}_j$ then $\mathsf{sync} \leftarrow 0$
    if $\mathsf{sync} = 1$ then $m \leftarrow \lightning$
    return $m$
```

Figure 2.9: Experiments to define INT-sfCTXT and IND-sfCCA security.

and initialise the states. The adversary $\mathcal{A}$ is then given access to a left-or-right encryption oracle $\mathsf{LoR}(\cdot)$, and a stateful decryption oracle $\mathsf{sfDec}(\cdot)$. This decryption oracle returns the decrypted ciphertexts only for out-of-sync queries, and returns $\lightning$ otherwise. The adversary's goal is to output a bit b', as its guess of the challenge bit b, and the experiment returns b' as well. We define the advantage of an adversary $\mathcal{A}$ as:

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

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

For a scheme to meet either of these notions, its decryption algorithm must be stateful. The 'sf' in the notion designations, which stands for 'stateful', can be attributed to the fact that the decryption and try oracles in the corresponding experiments are themselves stateful. A further contribution from the work of Bellare, Kohno, and Namprempre is to extend Theorem 2.2 to the stateful setting. This is stated formally in the following theorem.

Theorem 2.3: $\mathsf{IND\text{-}CPA} \wedge \mathsf{INT\text{-}sfCTXT} \longrightarrow \mathsf{IND\text{-}sfCCA}$. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For any* $\mathsf{IND\text{-}sfCCA}$ *adversary $\mathcal{A}_{sfcca}$ there exist adversaries $\mathcal{A}_{cpa}$ and $\mathcal{A}_{int}$ consuming similar resources to $\mathcal{A}_{cca}$ such that:*

$$\mathbf{Adv}^{\text{ind-sfcca}}_{\mathcal{SE}}(\mathcal{A}_{sfcca}) \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) + 2 \cdot \mathbf{Adv}^{\text{int-sfctxt}}_{\mathcal{SE}}(\mathcal{A}_{int})\,. \tag{2.2}$$

Chapter 3

Secure Network Protocols

Contents

This chapter introduces the necessary background on secure network protocols. We start with an overview of TCP/IP and a description of the three most used secure network protocols. We then look at some of the previous works that analysed the security of these protocols.

3.1 The TCP/IP Protocol Suite

At a very high level, the Internet can perhaps be best described as a network of computer networks. At its core is the TCP/IP protocol suite, which allows millions of computer networks based on different networking technologies to operate as one big computer network. Distinct computer networks are connected by means of gateways. A gateway is a special machine that has a physical network interface on more than one network, and can relay data from one network to the other. The

Internet Protocol (IP) is mainly responsible for relaying data across physical network boundaries. Central to its operation is the IP addressing scheme, which organises in a neat and hierarchical manner all the machines on the Internet. In essence every machine on the Internet is assigned an IP address that uniquely identifies the machine and its location. An IP address is structured hierarchically so that address prefixes identify sets of machines (i.e. logical networks) in which the machine is contained. For each of these logical networks there will be a router which is responsible for routing traffic between that set of machines and the rest of the Internet. Every logical network can in turn be split into smaller logical subnetworks for which other routers will be responsible. Internet data will commonly travel through several routers until it reaches its intended recipient. Thus the task of delivering data from one end to the other is distributed among several routers, each of which needs to have only a ‘partial view’ of the Internet.

3.1.1 IP

Data over the Internet travels in the form of IP packets. In order to travel within any physical network an IP packet must be encapsulated in a data frame that conforms to the link-layer technology of that network. When forwarding IP packets from one network to another, the gateway takes care of stripping off the old frame and encapsulating the IP packet in a new frame. An IP packet consists of an IP header followed by the data payload. Figure 3.1 depicts the format of an IP packet in more detail. Both the IP header and the data payload can vary in length as necessary. The Header Length field (HLEN) indicates the IP header length in 32-bit words whereas the Total Length field indicates the combined length of the payload and IP header in bytes. The source IP address field indicates the IP address of the machine from which the packet originates. Similarly the destination IP address field indicates the intended recipient of the IP packet, and is used by the intermediate routers to successfully deliver the packet. The first four bits of the header indicate the version of the IP protocol that was used to create the packet. This is the first field that any IP software will inspect in order to determine how to interpret the rest of the packet. To date IP Version 4 remains the most widely deployed version, but support for IP version 6 is growing in popularity. In this thesis we will restrict ourselves to IP version 4, and accordingly the packet format depicted in Figure 3.1 corresponds to this version.

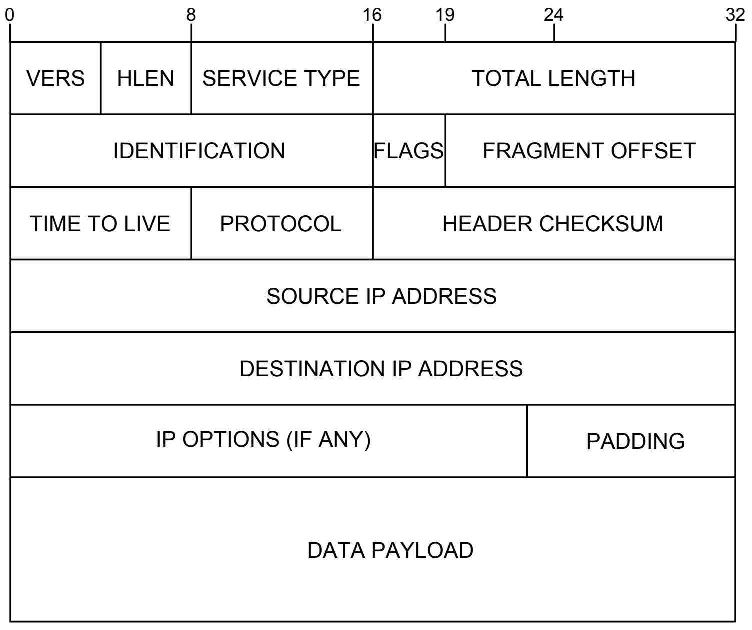

Figure 3.1: The structure of an IP packet.

The Protocol field contains a 8-bit value that indicates the type of data that is contained in the payload. Thus once the IP processing at the receiver is complete, this field indicates to which upper-layer protocol the payload should be forwarded to. Examples of such upper-layer protocols are TCP, UDP, and ICMP which we will describe shortly. The protocol number assignments are administered by a central authority: the Internet Assigned Numbers Authority (IANA). The Time To Live (TTL) field is essentially a counter that determines the life time of an IP packet. Typically the sender initialises this field to an integer value less than 256, and each intermediate router that forwards the packet decrements this counter by one. Thus the TTL fields acts as a 'hop count' rather than a timer in seconds as it was originally intended. When the counter reaches zero the router drops the packet and sends a 'time to live exceeded' (ICMP) error message to the sender. The Header Checksum field is used to detect errors in the IP header. The sender computes the checksum value by treating the header as a sequence of 16-bit integer values and adding them together using one's compliment arithmetic, and then taking the one's complement of the result. When computing the checksum, the Header Checksum field is assumed to contain zero.

In IP, errors are reported to communicating parties using the Internet Control Message Protocol (ICMP). ICMP specifies a separate format for ICMP messages which are then encapsulated in an IP packet. However, while ICMP uses the basic support of IP as if it were a higher level protocol, it is actually an integral part of IP that must be implemented by every IP module. ICMP messages contain a Type field and a Code field, both of which are a byte long, and together identify the exact error message. For instance the 'time to live exceeded' ICMP message is identified by type 11 and code 0. Similarly if the Protocol field in an IP packet indicates a protocol that is not supported by the receiver, a 'destination protocol unreachable' (type 3, code 2) ICMP message is returned to the sender. The contents of the ICMP payload depends on the type of ICMP message, but it will generally contain a portion of the IP packet that caused the error to occur. This is intended to help the sender better diagnose the cause of the error.

As we already mentioned, an IP packet may need to traverse different physical networks in order to reach its destination. Normally, physical networks can only carry data payloads below a certain size. This size limit is known as the Maximum Transmission Unit (MTU) of the network, and it varies according to the network's technology. This raises the question as to what happens when an IP packet is to traverse a physical network whose MTU is smaller than the IP packet size. Essentially the gateway in question will split the IP packets into a number of smaller IP packets. This mechanism is known as *fragmentation*, and accordingly the smaller packets are known as IP fragments. The IP fragments will then travel as independent IP packets, and will normally be reassembled by the receiver to recover the original IP packet. When a gateway fragments an IP packet into smaller fragments, the header contents of the original IP packet are copied into those of the smaller fragments. Three fields in the IP header, Identification, Flags, and Fragment Offset, control packet fragmentation and reassembly. The Identification field contains a unique integer that identifies the packet. This allows the receiver to identify fragments belonging to the same packet. The Fragment Offset field indicates the offset in the original packet of the data contained in the fragment. This is specified in eight-byte units, starting from offset zero, thereby enabling the receiver to reassemble the fragments in the correct order. The low-order bit of the Flags field is called the More Fragments (MF) bit, and it indicates whether the fragment is the last in the sequence. Thus if the MF bit is set or the Fragment Offset is not zero, it indicates that the IP packet is actually a fragment of a larger IP packet. Together the two fields also allow the receiver to determine whether all fragments belonging to a certain packet have been received. As soon as the first fragment reaches the receiver, a reassembly timer is started, and if it expires before all fragments are received, an ICMP ‘time exceeded’ (type 11, code 1) message is sent to the sender and the remaining fragments are discarded.

The TCP/IP protocol suite follows a *layered architecture* where one protocol builds on top of another. We have seen how IP builds on an aggregate of heterogeneous networks to yield a single uniform logical network. However the interface that IP provides is still rather rudimentary and is not immediately usable by applications. Accordingly a *transport protocol* that operates on top of IP is introduced to offer basic communication functionalities and present a simpler high-level interface to applications. Transport protocols are normally responsible for (a subset of) reliability, flow control, congestion control, multiplexing communication among different applications, and connection-oriented communication. The Transport Control Protocol (TCP), one of the core transport protocols in the TCP/IP protocol suite, offers all of these services. Transport protocols encapsulate application data into segments, which in turn are encapsulated in IP packets. TCP segments are also partitioned into a header and a data payload. We do not cover the TCP format in detail, but rather highlight the salient aspects which are relevant to this thesis.

3.1.2 TCP and UDP

The TCP protocol is a connection-oriented protocol, meaning that in order for two parties to communicate they first have to establish a connection. A connection provides applications with an interface similar to a data stream, as opposed to the datagram-oriented interface that IP provides. This means that an application does not have to deal with lost datagrams and sorting datagrams in the right order. Instead this is handled by TCP through sequence numbers in the segment headers, and its acknowledgement mechanism which handles the retransmission of segments. This is combined with a moderate error detection capability through a checksum field (similarly to IP) which is computed over the complete segment (rather than just the header). To each side of a TCP connection is associated a 16-bit port number which is reserved by the sending or receiving application. This port mechanism allows the multiplexing of multiple connections between the same pair of machines. Better still it allows any pair of communicating applications to share multiple connections. The TCP segment header includes two fields indicating the *source port* and the *destination port*. Thus the receiver can identify to which TCP connection a TCP segment belongs, from the sender and receiver IP addresses in the IP header, and the source and destination ports in the TCP segment header. Either party can terminate the connection at any point. This can happen gracefully, as is the case when all data has been sent and the connection is no longer needed, or otherwise in the event of an application error for instance, where the connection is torn down and any received data that has not yet been forwarded to the application is discarded.

The User Datagram Protocol (UDP) is another transport protocol which constitutes a core component of the TCP/IP protocol suite. It can be seen as a lightweight alternative to TCP for applications where a reliable service is not necessary. UDP offers an unreliable datagram-oriented application interface. It is unreliable in the sense that it does not handle the retransmission of lost or corrupted messages, and messages may be delivered out-of-order to the application. It is datagram-oriented in that communication takes the form of independent messages rather than a data stream as in connection-oriented communication. A port mechanism similar to that of TCP is employed to multiplex communication between applications.

The lower end of port numbers 0–1023 (TCP and UDP) is the *well-known* ports range which is reserved for specific services on which server applications listen for connection requests from clients. Typical examples are TCP port 80 for HTTP, TCP port 25 for SMTP, and UDP port 161 for SNMP. The range of port numbers from 1024 to 49151 are the registered ports. They are assigned by IANA for specific service upon application by a requesting entity. For instance, port number 1433 is reserved for Microsoft SQL and port 3690 is reserved for the SVN version control system. The range 49152-65535 contains dynamic or private ports that cannot be registered with IANA. This range is used for custom or temporary purposes, Client applications normally use source port numbers in this range for instance. If a TCP segment or UDP datagram is received with a destination port on which no application is listening, an ICMP ‘destination port unreachable’ (type 3, code 3) is returned to the sender.

Note that the TCP/IP protocol suite is a fairly elaborate set of protocols, and in this section we only covered basic aspects of it. The reader is referred to [28] for a more in-depth treatment, or the corresponding protocol specification documents known as RFCs, which are maintained by the Internet Engineering Taskforce (IETF).

3.2 Three Ubiquitous Security Protocols

We now turn our attention to security protocols intended to secure communication over the Internet and TCP/IP in general. Such protocols normally use an array of cryptographic primitives to provide a secure channel over which applications can communicate. We focus on three specific protocols, which are probably the most popular security protocols in use today.

3.2.1 TLS

As the name implies, Transport Layer Security (TLS), is a protocol for securing Internet communication at the transport layer of the TCP/IP stack. Its predecessor, Secure Sockets Layer (SSL), was developed by Netscape in the mid-1990s. In 1999, the protocol was adopted by the IETF and specified as TLS 1.0 in RFC 2246 [32]. It has since evolved through TLS 1.1 [33] to the current version TLS 1.2 [34]. Various other RFCs define additional TLS cryptographic algorithms and extensions.

TLS actually consists of four separate protocols: the Handshake Protocol, the Alert Protocol, the Change Cipher Spec Protocol, and the Record Protocol. We will be mainly concerned with the Record Protocol, which uses symmetric-key cryptographic primitives to yield a secure channel for use by the application layer. The other three protocols are collectively referred to as the TLS handshaking protocols. Among other things they are responsible for negotiating a ciphersuite (i.e. a set of cryptographic primitives), authenticating the parties involved to each other, establishing a shared secret from which key material is derived, and handling error messages. The Record Protocol uses a stateful variant of the authenticate-then-encrypt composition that we described in Section 2.4. The supported MAC algorithms are all HMAC-based, with MD5, SHA-1 and SHA-256 being the allowed hash algorithms in TLS 1.2 [34]. The encryption component can be either of the following: AES in CBC mode, 3DES in CBC mode, or the stream cipher RC4. The latest TLS specification [34] introduced support for dedicated authenticated encryption schemes, but as of the time of writing these have not yet received wide support in TLS implementations.

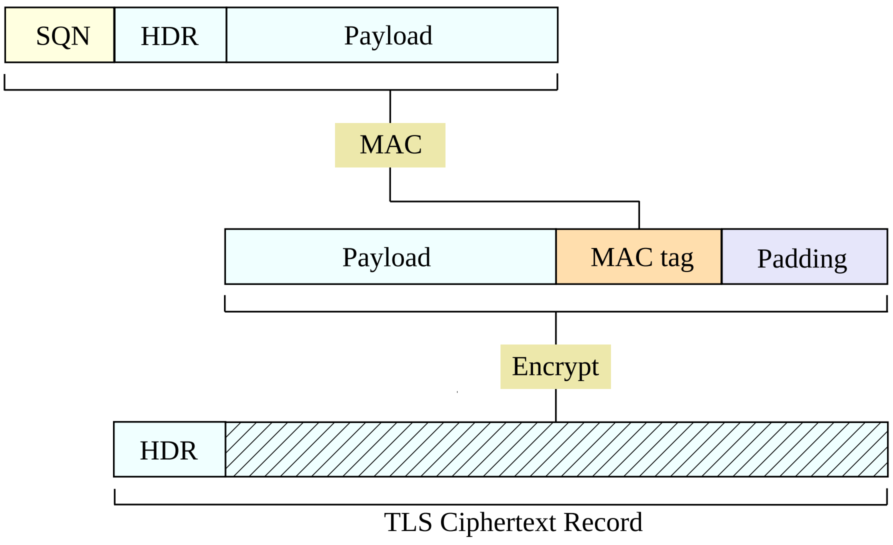

Figure 3.2: Cryptographic processing of a TLS record.

Data to be protected by TLS is received from the application and is fragmented into TLS plaintext records carrying data in chunks of 2^{14} bytes or less. A TLS record consists of a 5-byte header HDR, and a payload portion containing the application data. The record header consists of a 2-byte version field, a 1-byte type field, and a 2-byte length field. Before any cryptographic processing takes place the payload may be compressed, if this option was negotiated during the handshake phase. Figure 3.2 depicts the cryptographic processing that takes place on the TLS plaintext record to produce a TLS ciphertext record, which is then forwarded to the TCP layer for transmission. The sender maintains an 8-byte sequence number SQN which is incremented for each record sent. A MAC is then computed over the TLS record prepended with this 8-byte sequence number. The payload portion is then appended with the MAC tag, and if CBC encryption is to be used, this is in turn appended with a sequence of padding bytes. The length of the padding is such that the combined length of the payload, MAC tag, and padding is an integer multiple of the block size of the selected block cipher. The padding must consist of $p + 1$ copies of some byte value p, where $0 \leq p \leq 255$. Furthermore, at least one byte of padding must always be added (if CBC encryption is to be used). The padding may extend over multiple blocks, and receivers must support the removal of such extended padding. The resulting string, consisting of the concatenation of the payload, MAC tag, and possibly the padding, is then encrypted using CBC encryption or RC4. The transmitted TLS record is formed by prepended the resulting ciphertext with the header HDR. Note that the sequence number is not transmitted as part of the TLS record.

We conclude this section by briefly mentioning a variant of TLS known as Datagram Transport Layer Security (DTLS). The DTLS protocol is a developed from TLS by making minimal changes so as to allow it to operate over UDP instead of TCP. Its latest version, DTLS 1.2, which builds on TLS 1.2 [34], is defined in RFC 6347 [83].

3.2.2 SSH

Secure Shell (SSH) was originally designed as a secure alternative to remote login procedures such as *telnet*, *rlogin*, and *rsh*. It has since become a general purpose tool for securing Internet traffic. SSH was first created in 1995 by Tatu Ylönen a researcher at Helsinki University of Technology. Following a series of vulnerabilities that had been discovered, a major revision of SSH was released as SSHv2 by the IETF in a collection of RFCs [100, 98, 101, 99]. We here focus on SSHv2, and use SSH as a shorthand to refer to this version throughout this thesis.

The SSH protocol consists of three major components: the Transport Layer Protocol, the User Authentication Protocol, and the Connection Protocol. The Transport Layer Protocol operates over TCP, and it is responsible for the server host authentication and initial key exchange, as well as protecting the confidentiality and integrity of the subsequently transmitted data. The User Authentication Protocol operates over the the Transport Layer Protocol and is responsible for authenticating the user to the server. It offers different mechanisms for accomplishing this, such as password-based and public-key-based schemes. A single SSH connection may be shared among different applications. The Connection Protocol is responsible for multiplexing the secure channel, between the client and the server, into several logical channels. In this thesis we are mainly interested in the Binary Packet Protocol (BPP), the Transport Layer Protocol component responsible for authenticated encryption.

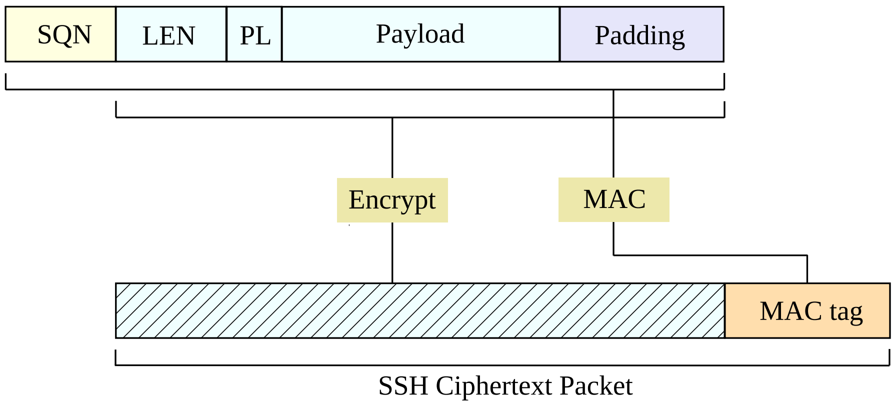

Figure 3.3: Cryptographic processing of an SSH BPP packet.

Figure 3.3 depicts the cryptographic processing employed by the Binary Packet Protocol. A payload message is first encoded by prepending a packet length field and padding length field, and appending some padding. The packet length field LEN is 4 bytes long and contains the total length (in bytes) of the encoded packet excluding the packet length field itself. The padding length field PL is 1 byte long and contains the total number of padding bytes. A minimum of 4 padding bytes must be added, up to a maximum of 255 bytes. Furthermore the padding should be random, and such that the encoded data ends on a block boundary. This encoded message is then encrypted, and a MAC tag is appended to the ciphertext to produce the final SSH Ciphertext Packet. The MAC value is computed over the concatenation of a 32-bit packet sequence number SQN, and the encoded (but not encrypted) message. The sequence number is set to zero at the start of an SSH connection, and is incremented after each packet. It is not sent over the channel but is maintained separately by both communicating parties.

The SSH RFC [101] mandates support for CBC using 3DES, recommends support for CBC using AES, and lists a further 12 block cipher variants in CBC mode as being optional. Only one optional stream cipher is listed, ARCFOUR. The RFC mandates that CBC mode with *initial packet chaining* be used throughout an SSH connection. In this variant of CBC mode, the last block of ciphertext from one packet is used as the IV for CBC mode for the next packet. Thus there is only one initial vector, and the packets on a connection form a single data stream. A later

RFC [8] defines a stateful version of counter mode encryption for use with SSH. For message authentication, it is required that HMAC using SHA1 be supported, with HMAC using MD5 also being listed as an option.

3.2.3 IPsec

Internet Protocol Security (IPsec) is a suite of protocols designed by the IETF. It offers security at the IP layer of the TCP/IP protocol stack, meaning that IPsec provides cryptographic protection for IP packets (or their payloads). IPsec was first specified in 1995, and the third and most recent version is specified in RFCs 4301–4309, released in 2005. It is deployed widely to build Virtual Private Networks (VPNs) and secure remote access solutions. In addition it is also a mandatory component of IPv6, and is part of the Universal Mobile Telecommunications System (UMTS) standard – used for securing the backbone network. The main constituent protocols of IPsec are the Authentication Header (AH), the Encapsulating Security Payload (ESP), and the Internet Key Exchange (IKE). Each protocol supports multiple configurations, and the three protocols can be combined in various ways. This high degree of configurability is what makes IPsec notoriously complex, and also makes it harder to analyse its security.

The AH protocol provides integrity protection, data origin authentication and anti-replay services for IP packets through the application of MAC algorithms and the inclusion of sequence numbers. The ESP protocol provides similar services to AH (though the coverage of integrity protection is more limited) and in addition provides confidentiality and traffic flow confidentiality services through symmetric key encryption and variable length padding of packets. Each of AH and ESP can be operated either in *transport mode* or in *tunnel mode*. Transport mode can be thought of as the basic way of using AH and ESP to protect IP packets. Tunnel mode processing can then be described as follows. The IP packet to be protected is first encapsulated in another IP packet, and AH or ESP (transport mode) processing is then applied to this new IP packet. Tunnel mode is typically used in VPNs, where the IPsec processing is applied by a gateway. In contrast, transport mode, is typically used in settings where end-to-end security is needed. The IPsec processing is thus applied by the host from which the IP packet originates. The IKE protocol

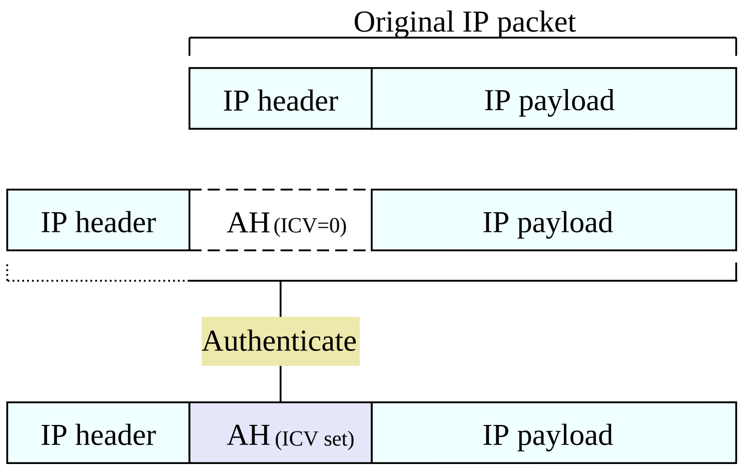

Figure 3.4: AH processing of an IP packet in transport mode.

employs a Diffie-Hellman protocol to exchange keys between IPsec parties. Alternatively keys can be set manually, which is normally simpler to configure in small networks. Another important component of every IPsec implementation is the Security Policy Database (SPD). This is a set of policies defining the processing rules for each type of IP traffic.

In this thesis we will be mainly concerned with the specifics of the AH and ESP protocols. We will assume that the necessary policies and key material are already in place. We now go onto describe these in more detail. For a more complete and yet accessible coverage of the cryptographic processing in IPsec, the interested reader can consult [77].

The AH protocol adds its cryptographic protection by inserting a bit sequence called the Authentication Header into IP packets. This is depicted in Figure 3.4 for transport mode. The exact format of the Authentication Header is shown in Figure 3.5. The Integrity Check Value (ICV) field contains the MAC tag used to authenticate the packet. The scope of the MAC calculation includes the IP payload, the IP header and the Authentication Header itself. Certain fields of the IP packet header, such as the TTL and checksum fields, cannot be input to the MAC calculation because they may change during the packet's transit across a network and so are unpredictable to the receiver. These mutable fields and the ICV field are both set to zero for the purposes of MAC calculation and verification. The length of the MAC tag depends on the particular MAC algorithm in use. Restrictions are that the MAC value must be an integral number of 32 bits in length and that the overall authentication header must be a multiple of 32 bits in length for IPv4. Typical MAC algorithms are HMAC-SHA1-96 [70] and AES-XCBC-MAC-96 [45], both having 96-bit long tags.

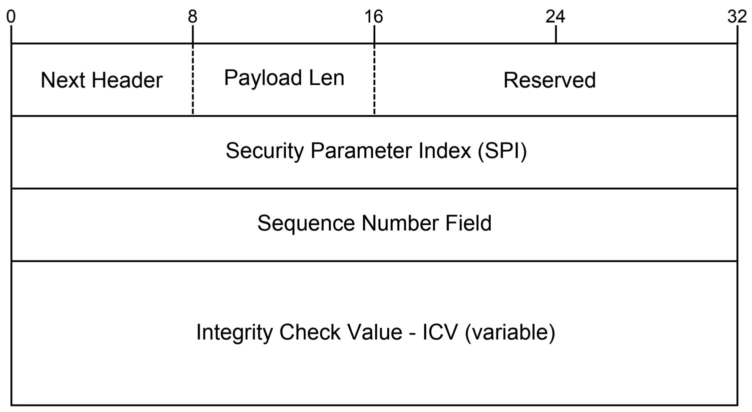
0
8
16
24
32
Next Header
Payload Len
Reserved
Security Parameter Index (SPI)
Sequence Number Field
Integrity Check Value - ICV (variable)

Figure 3.5: Authentication Header format according to RFC 4302 [61].

At the receiver, the MAC is checked and the packet discarded if the MAC is incorrect. In addition, when replay protection is enabled, the 32-bit sequence number carried by AH is compared to a sliding window of recently received sequence numbers. The packet is again rejected if the sequence number has already been received or if it is deemed to be too old by falling to the left of the current window. A packet having a valid MAC and a sequence number greater than the largest previously accepted will always be accepted, causing the window to be shifted to the right. RFC 4302 [61] also supports the use of 64-bit extended sequence numbers.

The Next Header field in AH is a one byte (8-bit) field indicating the type of the payload following the Authentication Header. For example, a value of 4 indicates that what follows is an IPv4 packet (as would be the case in tunnel mode), while a value of 6 indicates TCP. The Payload Length field indicates the length of the Authentication Header in 32-bit words, minus 2. It is needed because the ICV field may vary in length. The Security Parameters Index (SPI) field is a 32 bit value identifying the cryptographic parameters (such as the MAC key) that were used during outbound AH processing. The SPI is shared between sender and recipient, and allows the recipient to quickly obtain the cryptographic parameters necessary to perform inbound processing.

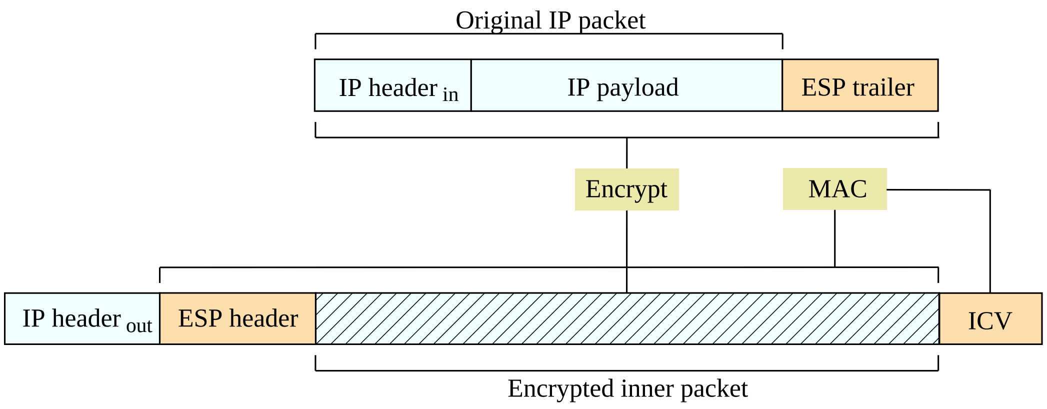

Figure 3.6: Encapsulating Security Payload processing of an IP packet in tunnel mode.

It is clear from the evolution of ESP that its primary purpose is to provide a confidentiality service. In its original version in [4], it only supported encryption with integrity protection coming from AH. In its second version [63] it was extended to support one or both (but not neither) of an encryption algorithm and a MAC algorithm. RFC 4303 extended ESP to additionally support combined mode algorithms (discussed in Section 2.4) that simultaneously provide confidentiality and authenticity.

In transport mode, an ESP trailer is first appended to the IP packet's payload and the result is encrypted. An ESP header is then inserted between the encrypted IP payload and the IP header. If the packet is to be integrity protected as well, a MAC is computed over the concatenation of the ESP header and the ciphertext, and the tag (ICV) is appended to the ciphertext. In tunnel mode, the original IP packet is first encapsulated in an outer IP packet, and this outer packet is then processed similarly. Note that in this case, since the inner packet constitutes the payload of the outer packet, the entire original packet is protected (including its header). The ESP processing of an IP packet in tunnel mode is illustrated in Figure 3.6, and Figure 3.7 shows the ESP format in more detail. Note, however, that Figure 3.7 does not apply for combined mode algorithms, which may not have explicit IV or ICV fields, for example.

The ESP header consists of the SPI and Sequence Number Fields, which are mainly used in the same way as in AH. However, if integrity protection is not enabled, the

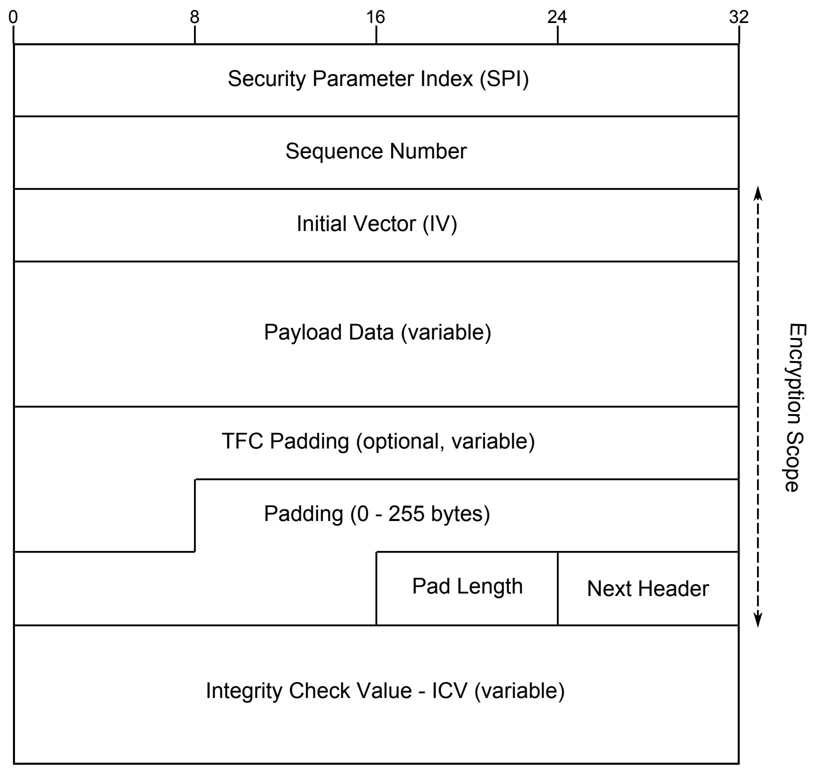

Figure 3.7: Encapsulating Security Payload format according to RFC 4303 [62].

sequence numbers must be ignored by the recipient. The ESP trailer serves mainly to ensure that when it is appended to the packet's payload, the combined length is an integer multiple of the block size. It consists of the padding, followed by a Pad Length field and a Next Header field. The Pad Length field indicates the number of padding bytes present, whereas the Next Header field serves the same purpose as in AH. It is permissible for the padding to be of variable length and to extend over multiple blocks. An ESP encryption algorithm may specify its own padding rule; otherwise a default rule is specified in [62, Section 2.4]. This default padding method is what is universally used in practice. The default padding consists of either a null string or t bytes of the form $1, 2, \ldots, t$ for some t with $1 \leq t \leq 255$.

RFC 4303 introduced two mechanisms for providing traffic flow confidentiality, that is, the provision of spurious traffic to frustrate an attacker's attempts to gather information from the mere existence of IPsec protected traffic, or from statistics concerning that traffic. These include the optional TFC padding and dummy packets. TFC padding can be inserted after the payload data. This is in addition to the normal ESP padding. However, TFC padding can only be used if the receiver is able to unambiguously remove it using information about the proper payload length that is embedded in the payload itself. This will be possible, for example, in tunnel mode, where the Total Length field in the inner packet header gives the needed information. In transport mode, this relies on the upper layer protocol format including a length field which can be used for the same purpose. Dummy packets can be indicated simply by using 59 for the protocol value in the Next Header field and otherwise creating a normal ESP header and trailer. According to RFC 4303, a receiver must discard any such packet without generating an error message.

RFC 4305 specifies TripleDES-CBC [82] and the NULL algorithm [46] as the mandatory encryption algorithms, and recommends support for AES-CBC [44] and AES-CTR [54]. As for integrity protection, it mandates support for HMAC-SHA1-96 [70] and the NULL algorithm, and recommends support for AES-XCBC-MAC-96 [45]. Other algorithms are allowed; a number of RFCs, such as [94, 55, 74, 66], specify additional algorithms that could be used with ESP. As already mentioned, RFC 4303 prohibits that the encryption algorithm and the integrity algorithm be both set to NULL. RFC 3686 states that if AES-CTR is used in ESP, then it must be accompanied by a non-NULL integrity protection algorithm. This is because AES-CTR is vulnerable to simple plaintext manipulation without some additional integrity protection. Note that RFC 3602 makes no such requirement for AES-CBC. This appears to be yet another incarnation of a long-standing issue in the IPsec RFCs [84, 20, 81, 29], in which CBC mode is perceived to offer some form of integrity protection, and is therefore less prone to plaintext manipulation (and related attacks).

3.3 Related Theoretical Analysis

Shortly after the work of Bellare and Namprempre on generic composition (discussed in Section 2.4), Krawczyk published a paper [69] titled ‘The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)’. Krawczyk analysed two instantiations of authenticate-then-encrypt, one where the encryption component is a one-time-pad and CBC encryption in the other.

He showed that these schemes satisfy IND-CPA security and a one-time variant of INT-CTXT in which the adversary is only allowed one query to the try oracle. In [13] it is shown that this one-time variant of INT-CTXT is equivalent to INT-CTXT up to a factor of q_t, where q_t is the number of try queries. It thus follows from Krawczyk's proof that these two instantiations are secure in the AE sense. However, in the case of CBC encryption, this result is of limited applicability to SSL/TLS. Krawczyk's analysis considers an instantiation in which the message must be a multiple of the block size, the tag length is equal to block size, and no padding is used. Unfortunately, this does not cover any usage case of TLS.

Maurer and Tackmann [73] analyse an authenticate-then-encode-then-encrypt composition, where the encoding step is meant to model any intermediate formatting – such as padding. They consider a *secure channel* formulation of security, within the framework of constructive cryptography [72]. They show that this composition, when instantiated with CBC encryption or the one-time-pad, yields a secure channel when the encoding step is a function. Thus their result only applies in the restricted case where only minimal padding is allowed.

A more accurate representation of the TLS Record Protocol was recently analysed by Paterson, Ristenpart and Shrimpton [78]. They consider a security notion called Length-Hiding Authenticated Encryption (LHAE), extending the AE notion with the intention to capture a scheme's limited ability of concealing plaintext lengths. Their results point out an interesting aspect of the authenticate-then-encode-then-encrypt composition as used in TLS with CBC encryption (MEE-TLS-CBC). More specifically they show that the tag size of the MAC, in relation to the block size, plays a crucial role in the security of the scheme. Their first contribution is a distinguishing attack against MEE-TLS-CBC that is successful whenever the tag size is smaller than the block size. The attack exploits variable length padding and is therefore outside the model of TLS used in [69, 73]. On the other hand, they show that this composition is LHAE secure as long as the sum of the minimum message length and the tag size is greater or equal to the block size. Note that in TLS the minimum message length is zero. An important assumption that is required for their proof is that decoding errors and MAC verification failures be indistinguishable to the adversary.

An analysis of the SSH Binary Packet Protocol was given by Bellare, Kohno and Namprempre in [16]. They consider several variants of the SSH BPP. They denote by SSH-IPC (SSH with interpacket chaining) the SSH BPP using CBC mode as defined in [101]. They present a chosen-plaintext attack against SSH-IPC, which is attributed to Wei Dai. Next they consider SSH-NPC, which refers to the SSH BPP using CBC mode without packet chaining, using a fresh, random IV for each packet, but with a fixed padding format. While this variant is IND-CPA secure, it is not secure in the IND-CCA sense, as it is vulnerable to a reaction attack[1]. They then go on to propose three variants SSH-$NPC, SSH-CTRIV-CBC and SSH-CTR, which they prove secure in the sense of IND-CPA and INT-sfCTXT, and hence by Theorem 2.3 is also IND-sfCCA secure. SSH-$NPC refers to the SSH BPP using CBC mode with random per packet IVs and random padding. SSH-CTRIV-CBC refers to using CBC mode with IVs generated by encrypting a counter. In this proposal, the IVs are not transmitted, and the encryption and decryption are stateful. SSH-CTR refers to the SSH-BPP with counter mode encryption, where the counter is maintained by the sender and the receiver rather than being transmitted at the start of the packet.

As we shall see later on, the analysis of Bellare, Kohno, and Namprempre misses an important practical aspect of the SSH protocol. This relates to the fact that over the Internet, an SSH ciphertext packet may be delivered in a piecewise fashion. This gives rise to a new line of attack which turns out to be fatal for the security of SSH-$NPC [1]. We will give more details about this attack in Section 3.5. Paterson and Watson [79] look at SSH-CTR as implemented in the OpenSSH distribution. They prove its security in a model which grants the adversary the ability to deliver ciphertexts in a piecewise fashion, thereby showing that it is immune to the attack in [1].

IPsec has not received much formal treatment, probably in part due to its higher degree of complexity. As we saw in the previous section, when integrity protection and confidentiality are both enabled in ESP, they are combined in an encrypt-then-authenticate fashion. Alternatively, it is also possible to compose AH and ESP both in encrypt-then-authenticate and authenticate-then-encrypt modes. Encrypt-

[1]In a reaction attack an attacker is able to gain information about a target ciphertext, solely from learning whether other related but distinct ciphertexts are valid or not.

then-authenticate was shown in [17] to always yield AE security (if the constituent primitives meet standard security requirements), albeit the composition they consider omits many specifics of the IPsec encrypt-then-authenticate realisations. As for authenticate-then-encrypt, none of the aforementioned analyses are relevant to the IPsec setting. The analysis of [69] omits any intermediate encoding steps, and it does not consider tag sizes that match the ones used in IPsec. As explained in Section 3.2.3, the paddings used in IPsec are not required to be minimal, which means that the results of [73] also do not apply. As for the analysis in [78], even though they consider non-minimal padding, their results are specific to TLS and the padding format it uses. In Chapter 4 we will present new results on the security of authenticate-then-encrypt in IPsec. Finally, the IPsec RFCs [62] also allow ESP to be operated with CBC mode encryption only, i.e. without any additional integrity protection. As early as 1996, Bellovin [20] had sketched a number of attacks to highlight the problems of permitting such a configuration. Nonetheless these attacks omitted many practical details and their practicality remains questionable. Later in [81], Paterson and Yau discovered and implemented a series of plaintext-recovery attacks against the Linux IPsec implementation. However, the Linux IPsec implementation omitted certain policy checks that were mandated by the RFCs which would have prevented these attacks. Thus their attack did not pose a threat to any RFC-compliant IPsec implementation, and was perceived as highlighting a vulnerability in Linux rather than in the IPsec standard. The matter was later settled in [29] by Degabriele and Paterson, who discovered and implemented full-plaintext recovery attacks against the OpenSolaris IPsec implementation, which does conform to the IPsec RFCs. Their attack exploits padding oracles, which we discuss in the next section. Despite such incontestable evidence on the insecurity of ESP in encryption-only, to this day, this configuration is still permitted by the IPsec RFCs.

3.4 Padding Oracle Attacks

Padding oracle attacks were first introduced by Vaudenay in [93], and are mostly specific to CBC encryption[2]. As we saw earlier, in practice data needs to be padded

[2]It is possible to extend these attacks to similar modes of operation, such as HCBC [11], but that is beyond our scope.

according to some rule before it can be encrypted using CBC mode. Many padding schemes that are used in practice are susceptible to padding oracle attacks. Padding oracle attacks assume a setting in which the attacker has access to a special oracle $\mathcal{O}(\cdot)$ that reveals whether a ciphertext is correctly padded or not. This attack model was first considered by Bleichenbacher [21], in the public-key setting, to attack PKCS#1 v1.5 a padding scheme suited to RSA encryption. Vaudenay showed that for many widely-used padding schemes, a padding oracle can be leveraged by an attacker to decrypt CBC-encrypted ciphertexts, using a relatively small number of queries to the oracle.

For concreteness consider the padding scheme employed in TLS. Here, a correctly padded ciphertext means that the CBC decryption of the ciphertext is a byte string ending in one of the valid padding patterns "0x00", "0x00 $\|$ 0x01", etc. Let c^* be the target ciphertext that an attacker wishes to decrypt, and denote by c_i^* the i^{th} block of this ciphertext. For any ciphertext block c, we index its bytes by c_j for $0 \geq j \leq b-1$ starting with the leftmost byte, where b is the block size in bytes. Vaudenay's padding oracle attack proceeds as follows. The attacker forges a new ciphertext $r \| c_i^*$ by appending the target ciphertext block c_i^* to a block r that it picks uniformly at random. It then submits this ciphertext to the padding oracle. Due to the block r being random, the decryption of this ciphertext will yield a random block of plaintext. As such, it is unlikely that this plaintext block will be correctly padded. However, in the event that it is, the most likely case is when the last byte is "0x00". The probability of this event occurring is 2^{-8}. The attacker can however iterate through all possible values of this last byte by iterating through all possible values of $r[b-1]$. When a value for $r[b-1]$ is found such that $\mathcal{O}(r \| c_i^*)$ returns valid, it means that $r[b-1] \oplus \mathcal{D}_K(c_i^*)[b-1] =$ "0x00". Next the attacker aims to find a ciphertext that terminates with the padding "0x01 $\|$ 0x01". It thus sets $r[b-1]$ such that the last byte of the decrypted ciphertext is "0x01", and iterates through all possible values of $r[b-2]$ until the padding oracle returns valid again. The attacker continues this process, extending the padding byte by byte, until the padding fills the complete block. At this point it knows that $r \oplus \mathcal{D}_K(c_i^*) =$ "0x$(b-1)$ $\| \ldots \|$ 0x$(b-1)$". Since it knows r, it can compute the block $\mathcal{D}_K(c_i^*)$. It can then recover the plaintext block corresponding to c_i^* by computing $\mathcal{D}_K(c_i^*) \oplus c_{i-1}^*$. All other blocks in the target ciphertext can be recovered similarly, requiring at most $b \times 2^8$ oracle queries per block.

In the above we have assumed that the first validly padded ciphertext that is found by the attacker corresponds to the shortest padding pattern. The odd cases where this does not hold are easy to detect, and are in fact desirable as they reduce the overall complexity of the attack. In addition it is easy to see that the attack can be easily adapted to other padding schemes of similar formats. What remains to address is how to realise the padding oracle itself. In [93] Vaudenay conjectured that padding oracles could possibly be realised for TLS, IPsec and SSH. In the case of TLS for instance, he outlined how error messages from a TLS server could be used to leak the validity of padding in ciphertexts. Such errors however are fatal, in that they result in the TLS connection being torn down, which prevents the above attack. Thus the attack as described in [93] remains theoretical in nature. Later works [27, 29, 30, 37, 57, 2, 3] managed to overcome such practical limitations, and realised variants of Vaudenay's attack against practical cryptosystems such as TLS, IPsec, ASP.NET, XML encryption and DTLS. We will not cover the details of these attacks at this point, but we will return back to them later on in this thesis. In particular we will cover in depth some attacks against IPsec that were published in [30] in Chapter 4.

3.5 A Ciphertext Fragmentation Attack on SSH

We conclude this chapter by presenting an attack against SSH due to Albrecht, Paterson and Watson [1]. The attack works only when SSH is used with CBC mode encryption, and can recover up to 32 bits of plaintext from any ciphertext block. In order to understand the attack, let us first consider how decryption takes place in the SSH BPP. Remember that over a TCP/IP network, a BPP packet may be fragmented and delivered in a piecewise fashion. Since there is no length indicator for a BPP packet other than the content of the packet length field, any SSH implementation must decrypt the first ciphertext block to obtain that field and use it to determine how much data to accept before deciding that a complete BPP packet has arrived and moving on to perform the MAC check. Thus an SSH implementation will await further data, unless sufficient data has already arrived to complete the packet.

An attacker can exploit this step in the decryption process as follows. Given any target ciphertext block c_i^*, an attacker can obtain the first 4 bytes of the corresponding plaintext block. Let c_{i-1}^* denote the block preceding c_i^* in the ciphertext stream, and let c_ℓ denote the last ciphertext block transmitted. The attacker sends to the receiver as a first fragment of ciphertext the block c_i^*, so that it is interpreted as the block containing the length field. Then the attacker feeds a sequence of random 1-block fragments to the receiver, until an (encrypted) error message is returned and the SSH session is terminated. At this point, the attacker knows that the MAC has been checked, and so knows that the complete ciphertext has been received by the decryption oracle. The number of 1-block fragments needed to trigger this event, then, reveals the value of the length field which corresponds to the first 4 bytes of $\mathcal{D}_K(c_i^*) \oplus c_\ell$. The attacker can then use this to compute the first 4 bytes of the target plaintext block $\mathcal{D}_K(c_i^*) \oplus c_{i-1}^*$.

The above exposition omits a number of important details; what happens for instance if the length field in $\mathcal{D}_K(c_i^*) \oplus c_\ell$ is not a multiple of the block length? Upon recovering the length field, the decryption algorithm performs a number of checks to verify its validity. The OpenSSH implementation for instance would check[3] that the length field is not less than 5 or greater than 2^{18}. The latter bound is intended to mitigate against certain Denial-of-Service (DoS) attacks that relate to ciphertext fragmentation[4]. If this check fails the session is terminated and an error message is sent over the connection. The OpenSSH implementation then verifies that the total number of bytes expected in the packet is a multiple of the block size. This check is also fatal, in that if it fails the SSH session is terminated, but no error message is sent. However the attacker can detect such an event, as it will result in a termination of the TCP connection over which the SSH session is running. Thus, if after injecting c_i^*, the TCP connection over which the SSH connection is running is terminated without an SSH error message (indicating a failure of the second length check) or the SSH connection enters a state in which it is waiting for more data, then the attacker knows that the decrypted block has passed the first length check. This implies that the first 14 bits of the decrypted block are all zero. At this point the attacker can already recover the first 14 bits, and it succeeds with probability $2^{-14} - 5/2^{18} \approx 2^{-14}$. If the SSH connection enters a wait state, then both checks have passed and the attacker can continue as described above by injecting 1-block

[3]After that the attack in [1] was reported, these validation checks have been amended in later versions of OpenSSH.

[4]We will elaborate more on such DoS attacks in Chapter 6.

fragments until a MAC-related error message is triggered. This allows the attacker to recover all 4 bytes, but the attack succeeds only with a probability of roughly 2^{-18} (assuming a block size of 128 bits).

Chapter 4

New Attacks on IPsec

Contents

In this chapter we present new attacks against authenticate-then-encrypt configurations of IPsec. We elaborate on the practical details and our experience in implementing the attacks.

4.1 Introduction

IPsec is a notoriously complex protocol suite, but one of great importance in today's Internet. Part of IPsec's complexity arises from a deliberate attempt by IPsec's designers to provide a flexible and highly configurable approach to providing security services for IP traffic. The RFCs specifying the major component protocols ESP, AH, IKE [61, 62, 60] and that describing the IPsec architecture [65] offer only limited guidance to end users about how best to configure IPsec to achieve their desired security goals. Moreover, little security analysis of IPsec seems to have been published. In particular, whilst it is by now well-established that using ESP in encryption-only configurations is insecure in general [20, 81, 29], there appears to have been no *systematic* security evaluation of the many different ways of combining encryption and integrity protection that are allowed by IPsec:

- ESP may provide its own integrity protection, in which case it is provided by a MAC algorithm that is applied after ESP's encryption – an encrypt-then-authenticate construction.
- Alternatively, AH can be used to provide the protection, again using a MAC algorithm, though with the MAC algorithm having a greater scope than in ESP. In this case, packets may be first integrity protected by AH and then encrypted using ESP, or first encrypted by ESP and then integrity protected by AH (where now the extended scope of AH's integrity protection means that more fields of the IP header are protected than would be the case with ESP-provided integrity protection).
- It is even possible to achieve an authenticate-then-encrypt construction using two layers of ESP processing.
- Further, the current version of ESP allows combined-mode algorithms to be used, wherein encryption and integrity protection are rolled into a single processing step.
- In all of the above configurations, AH and/or ESP may each be applied in either tunnel mode or transport mode.
- To add a final dimension, both AH and ESP allow sequence number checking to be performed as an option, in order to provide protection against replay attacks. This replay protection service should be disabled if manual keying is used (see [61, Section 5] and [62, Section 3.3.3]), is recommended to be disabled for multicast traffic ([62, Section 3.4.3]), and may be problematic when differentiated classes of traffic are protected by a single SA ([65, Section 4.1]). As we shall see, whether the replay protection service is disabled or not has a significant impact on some of our attacks.

It is notable that the previous version of the IPsec architecture [64] was *more* specific about which combinations must or must not be supported in IPsec implementations than is the current version [65]: the former required support for some basic configurations and explicitly outlawed the combination of AH followed by ESP both in transport mode, while the latter makes no prohibitions.

What guidance can be extracted from the literature? Theoretical support for the encrypt-then-MAC options comes from [17, 69], where it is shown that this approach generically provides IND-CCA security if the component encryption algorithm is IND-CPA secure (as is the case, for example, for CBC mode encryption with a random IV — see Section 2.3.3) and the component MAC algorithm is strongly unforgeable.

Concerning authenticate-then-encrypt options, it is noted in [65] that *"an underlying integrity service, such as AH, applied before encryption does not necessarily protect the encryption-only confidentiality against active attackers"*, suggesting that such configurations should be avoided. Here, [65] cites [69] for theoretical support. However, a closer examination of [69] shows that it contains positive security results about the authenticate-then-encrypt construction when the encryption scheme is implemented using either a secure stream cipher or CBC mode of a block cipher. These are the primary encryption schemes currently supported by IPsec standards. Moreover, the known examples in [17, 69] showing that authenticate-then-encrypt constructions are not generically secure are rather artificial. Thus the results of [69] could be interpreted as providing support for authenticate-then-encrypt configurations of IPsec. Further support comes from a widely-cited critique of IPsec by Schneier and Ferguson [39], which states *"When both encryption and authentication are provided, IPsec performs the encryption first, and authenticates the ciphertext. In our opinion this is the wrong order"* and later goes on to say *"The ordering of encryption and authentication in IPsec is dangerous."* In [39] the argument is made that a protocol should authenticate what was meant, not what was said, with SSL as analysed in [95] being given as an example of a protocol adopting the "correct" approach of authenticate-then-encrypt. Moreover, a putative attack against encrypt-then-authenticate configurations of IPsec is given in [39], lending further support to the authenticate-then-encrypt choice for IPsec[1]. A standard textbook on network

[1]However this attack requires the receiver to use the wrong key when decrypting, and it is hard to envisage the circumstances under which this could occur in IPsec, except perhaps with re-use of SPIs in a manually-keyed deployment.

security [92] discusses several benefits that accrue from using an authenticate-then-encrypt configuration of IPsec, including the ability to store MAC values along with plaintexts for later checking. A textbook aimed at implementers of cryptography [40] extensively discusses the merits and demerits of the MAC-then-encrypt approach to building secure channels, and eventually recommends this construction over other choices.

In summary there seems to be no solid argument about the security of authenticate-then-encrypt configurations in IPsec. Theoretical results appear to be 'too coarse' to be meaningful for the case of IPsec, while all claims found in the literature that are specific to IPsec are heuristic in nature. In addition, given the arguments on both sides, and in the absence of firm guidance from the RFCs or other sources, it seems plausible that a network administrator might well be tempted into selecting a authenticate-then-encrypt configuration of IPsec. In what follows we settle this matter in the negative. We present a series of practical attacks against *all* authenticate-then-encrypt IPsec configurations.

4.2 Preliminaries

Our attacks build on Vaudenay's padding oracle attack [93], and exploit the interplay between IP and IPsec in order to realise the padding oracle. For concreteness, we study the common use case of using IPsec to build a simple site-to-site VPN, such as the one illustrated in Figure 4.1. We assume that all cryptographic processing is carried out at a pair of security gateways, but our attacks also extend to situations where AH processing is carried out at hosts behind the gateways. Our attacks come in three basic flavours, each with two main variants depending on whether IPsec's optional replay protection is enabled or not. Our attacks are powerful in the sense that they can be used to recover plaintext from arbitrary IPsec-protected packets. But they each have different characteristics in terms of their complexity, their requirements for the attacker's degree of control over the network, and their plaintext requirements.

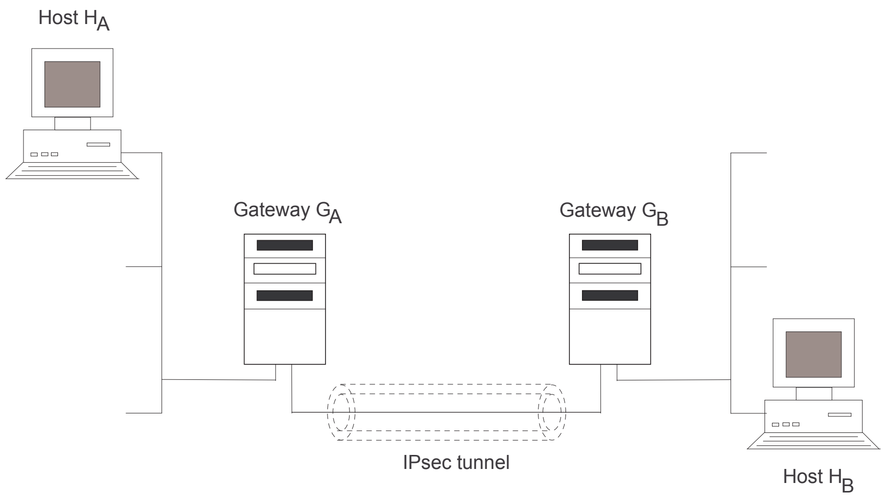

Figure 4.1: Network set-up.

4.2.1 IP and IPsec

Introductions to IP and IPsec, sufficient to understand the material in this chapter, were presented in Section 3.1 and in Section 3.2.3 respectively. We now highlight some of the finer details that relate to our attacks. Our exposition of the attacks will initially assume an authenticate-then-encrypt configuration in which AH is first applied in transport mode, followed by (encryption-only) ESP in tunnel mode. The resulting packet format is illustrated in Figure 4.2. We will then elaborate on how the attacks can be extended to the other configurations. Throughout we will assume that CBC encryption is in use. A modification of our attacks would work against AES-CTR, if it were not for the fact that [54] specifying AES-CTR requires that it must be used in combination with ESP-provided integrity protection, implicitly in an encrypt-then-authenticate construction. For ease of exposition we assume 64-bit extended sequence numbers are not selected, but our attacks still work if they are.

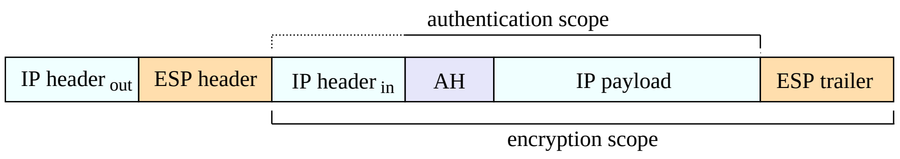

Figure 4.2: IPsec packet format assumed in the exposition of our attacks.

In our attacks we assume that the relevant RFCs have been carefully followed by an implementer. For example, our attacks exploit the recommendation of the ESP RFC [62] to perform full padding checks when decrypting, and two of them rely on support for Traffic Flow Confidentiality (TFC) padding that is mandated in [62]. One of the attacks depends on the details of IPsec's treatment of fragmented packets, while all depend on the manner in which IPsec handles ICMP traffic. Our attacks are developed with the RFC specifications in mind, but previous work [29] has shown that IPsec implementations do deviate significantly from the RFCs in ways that can stop attacks from working in practice. To compensate for this, we report on the experimental validation of our attacks against the OpenSolaris implementation of IPsec, showing that two out of three of our attacks 'on paper' can be converted into working attacks against a real implementation. Our choice of OpenSolaris was driven by the high quality of its code and its close adherence to the IPsec RFCs, and not because it has any particular weaknesses that we wanted to exploit. We believe that our attacks would apply to any comparably careful implementation of IPsec.

The IPsec architectural RFC [65] explains in detail how IPsec should handle ICMP messages, distinguishing between error and non-error messages. Our attacks use ICMP messages of both types, and the specific messages used in our attacks are not blocked by IPsec. However, they are only visible to the attacker in encrypted form and so typically need to be detected by their characteristic (though implementation-dependent) lengths, or via timing correlation.

4.2.2 Mauling IPsec-Protected Packets

In our attacks, we will flip certain bits in the headers of inner datagrams, by flipping bits in the initial vectors of ciphertexts (explained in Section 2.3.3). Any such modifications will require further compensation to be made elsewhere in the header so that the Header Checksum (calculated as the 1's complement of the 1's complement sum of the 16-bit words in the IP header) is still correct – otherwise the inner datagram will be silently dropped. In [29, 81], a number of techniques were developed for 'correcting' checksums in an efficient manner. We need to further develop these techniques so that our attacks are efficient for the IPsec configurations considered here.

Considering each 16-bit field in the IP header as an unsigned integer, suppose we wish to subtract the value δ from one of these 16-bit fields. Let S represent the 1's complement sum of all the 16-bit fields over which the checksum is computed, then the IP header checksum is given by $\overline{S}$ (the complement of S). Thus the new value of the IP header checksum should be set to $\overline{(S \boxplus \overline{\delta})}$ where $\boxplus$ denotes 1's complement addition. Then we need to select a 16-bit value `mask` such that:

$$\texttt{mask} \oplus \overline{S} = \overline{(S \boxplus \overline{\delta})}$$

and XOR this value `mask` to the appropriate field in the IV. We can rewrite this equation as:

$$\texttt{mask} = \overline{S} \oplus \overline{(S \boxplus \overline{\delta})} = S \oplus (S \boxplus \overline{\delta}).$$

Hence we can, for a fixed value of δ, compute all possible solutions `mask` to the above equation along with their probabilities of success in correcting the checksum, assuming that S is a uniformly distributed 16-bit value. We then use the list of possible values `mask` in order of decreasing probability when trying to correct the checksum.

An example is in order. Suppose we wish to decrease the TTL field from a known value `0xFF` to the value `0x00` and correct the checksum. Because of the position of the TTL field in the IP header, this implies a 16-bit value $\delta =$ `0xFF00`. Some of the resulting 66 masks having non-zero probability are shown in Table 4.1 along with their probabilities, which were calculated by exhaustive search over S. In this case, the number of trials required is decreased from the average of 2^{15} that would be needed using the methods of [29, 81] to an average of only 6.75. On the other hand, assuming nothing about the TTL field except that it is uniformly distributed, then a simple calculation using a variant of this approach shows that the expected number of trials needed to set the TTL field to `0x00` and correct the checksum is only 382.

This idea can be combined with the idea from [29] of using the ID field to compensate for the bit flips, rather than the checksum field itself. Because of the location of this field in the second 32-bit word of the IP header, this allows the above improvements to be deployed even for a 64-bit block cipher.

mask	probability
0000 0001 0000 0001	2^{-2}
0000 0011 0000 0001	2^{-3}
0000 0001 0000 0011	2^{-3}
0000 0111 0000 0001	2^{-4}
0000 0011 0000 0011	2^{-4}
0000 0001 0000 0111	2^{-4}
0000 0001 0000 1111	2^{-5}
$\vdots$	$\vdots$
1111 1111 1111 1111	2^{-16}

Table 4.1: Table of masks and probabilities for $\delta = 0xFF00$

4.2.3 ESP Trailer Oracles

Our attacks will make use of ESP trailer oracles, a concept introduced in [29] as an extension of the padding oracle concept from [93]. Such an oracle tells the attacker whether or not the trailer fields (including padding, pad length and NH bytes) of an encryption-only ESP-protected packet are correctly formatted — see Section 3.2.3 for the ESP trailer format. It is shown in [29] how repeated access to such an oracle allows an attacker to decrypt ESP-protected ciphertext blocks in a byte-by-byte fashion, at a cost of at most 2^{16} queries to the oracle to extract the rightmost 2 bytes of the target block and at most 2^8 queries for each remaining byte of the target block. We next outline how the ideas from Section 3.4 can be adapted to the case of IPsec.

Suppose we have a *carrier packet* that is protected by encryption-only ESP in tunnel mode, and a target ciphertext block c_i^* (from any packet protected by the same key K). The rightmost 2 bytes of c_i^* are extracted as follows. We splice blocks r, c_i^* onto the end of the carrier packet, and submit this new packet to the oracle. Here r is a randomly selected block. By varying the rightmost 2 bytes of r in a systematic fashion, we can explore all possible values of the rightmost 2 plaintext bytes in the block $r \oplus \mathcal{D}_K(c_i^*)$; these are interpreted as the Pad Length and Next Header bytes of the ESP trailer by the oracle, and it is argued in [29] that, with high probability, only the values `00,04` for these bytes will produce a positive response from the oracle. Once the oracle responds positively, the corresponding original plaintext bytes from $c_{i-1} \oplus \mathcal{D}_K(c_i^*)$ can be easily recovered by simple XOR arithmetic. The attack is then extended to plaintext bytes further to the left in the block by trying to construct longer valid trailer byte patterns, starting with `01,01,04`.

This leaves the question of how to construct an ESP trailer oracle. This problem was solved in [29] for the encryption-only case by constructing a special packet that, providing the packet was not dropped because of a failure of ESP's padding checks, always generated some kind of error response. Usually this takes the form of an ICMP message. In [29], for tunnel mode, encryption only ESP, this was done by using CBC bit flipping and checksum correction to create a packet whose inner packet had an unsupported protocol field. The resulting ICMP message is usually transmitted in encrypted form on the IPsec tunnel, but it was shown in [29] how such messages can be detected based on characteristic lengths or via timing correlation.

In summary, to mount this kind of attack, we need a carrier packet that produces a detectable response whenever ESP's trailer formatting checks pass. In [29], this required modification of IP header fields in the inner packet. This clearly creates a problem when the inner packet is protected by AH, as it is in the situations we are interested in here: now modifications to header fields may be detected by AH processing and the packets dropped, causing the oracle to be lost. An extra level of complication arises if AH's replay protection is enabled: now, each carefully-constructed carrier packet can only be used once, since if it were to be repeated, its inner packet would be deemed to be a replay during AH processing and so dropped, again causing the ESP trailer oracle to be lost. Finally, we also want to consider transport mode configurations of IPsec, and additional ideas are needed to cater for this.

As we explain in the sections that follow, all of these problems can be overcome and appropriate ESP trailer oracles constructed.

4.3 The Attacks

We begin by describing our three basic attack ideas in the context of the IPsec configuration depicted in Figure 4.2 that first applies AH in transport mode and then ESP (encryption only) in tunnel mode to packets flowing from G_A to G_B. This seems to us to be the most natural MAC-then-encrypt configuration, and it also turns out to be the easiest to attack. We then go on to explain how to extend the attacks to other MAC-then-encrypt configurations. In each case, we explain how to recover the plaintext block corresponding to a single target ciphertext block c_i^*. Of course, all of the attacks extend to multiple blocks in the obvious way.

4.3.1 Attack 1: A Chosen Plaintext Attack

Our first attack requires a single chosen plaintext and can recover arbitrary IPsec-protected plaintext. The attack exploits the fact that neither TFC padding nor ESP's normal encryption padding are protected by AH's MAC, and that, in accordance with [62], these bytes are discarded by the receiver before the inner packet is passed to AH.

Suppose for now that AH replay protection is disabled, and recall that ESP replay protection will always be disabled in this configuration. Suppose the attacker has available a single IPsec-protected packet of the form depicted in Figure 4.2, for which the inner IP packet has as its payload an ICMP echo request, which can be directed either to the gateway G_B itself or to a host behind that gateway. Clearly, if this packet is injected into the network towards G_B, we will see an (encrypted) ICMP echo reply message in the reverse direction on the VPN between G_A and G_B. Moreover, because AH and ESP sequence number checking is disabled, this packet, if repeatedly injected into the network, will always cause such a response. This packet can be used directly as a carrier in an ESP trailer oracle attack, as described in Section 4.2.3. Here, ESP's handling of TFC bytes ensures that the inner packet presented to AH after ESP processing at G_B always passes AH's MAC check, even after the blocks r, c_i^* have been spliced onto the carrier packet. This is because after the ESP trailer is checked and removed, any remaining plaintext resulting from the spliced blocks together with the original ESP trailer will be interpreted as TFC padding and discarded. Moreover, none of these discarded bytes are covered by AH's MAC. So, with a single chosen plaintext and an average of slightly more than 2^{15} trial packet injections, any complete block of plaintext can be recovered.

This attack applies no matter what are the key-size and block-size of ESP's encryption algorithm. It can also be applied if the inner IP packet carries TCP instead of ICMP: now every received TCP segment provokes a TCP ACK packet of some type in response, so every modified carrier packet that passes ESP processing at G_B will generate a detectable message in the reverse direction. Even if the TCP connection for the TCP segment in the carrier packet is already closed, a TCP RST packet will be sent in response, so our attacker will always get the response he requires. This applies whether the endpoints for the TCP connection are the gateways themselves or hosts behind these gateways. Assuming that the inner IP payload carries a TCP message is a mild chosen plaintext assumption. This can be replaced by an even weaker assumption by simply observing packets to see which ones generate replies, and then using one of those packets as the carrier packet.

Attack 1 with AH replay protection enabled We can extend the above attack to the case where sequence number checking is on. The attacker first gathers, for each byte (or pair of bytes in case of the rightmost bytes) of plaintext that he wishes to extract, a packet that is expected to generate a reply. These packets might carry ICMP or TCP, for example. We make the assumption that the attacker can put these carrier packets in order of (roughly) increasing AH sequence number. This is reasonable, since they are likely to be intercepted in such an order. The attacker also needs to control the flow of packets on the network so that the sequence numbers in his carrier packets are always seen as being 'fresh' during any AH processing at G_B for the duration of his attack. This can be achieved by firstly blocking all other packets from G_A to G_B except the attacker's carrier packets during the attack, and secondly by switching to the next carrier packet each time a response packet is detected on the VPN between G_A and G_B. The latter step coupled with our assumption about AH sequence number ordering ensures that, each time ESP trailer processing completes and AH processing is done, packets are not rejected by AH because they have repeated (or old) sequence numbers. Otherwise, the attack is as before.

For ease of presentation, we have described a simple version of the attack that requires the attacker to control the flow of traffic during the attack. It can be adapted to be less disruptive to traffic flow by making use of carrier packets as they become available to the attacker, but this would be more complex to implement.

The only drawbacks of Attack 1 are its very mild assumptions about the nature of plaintexts, its consumption of multiple carrier packets when AH replay protection is enabled, and the complexity of implementing the attack in a non-disruptive manner in this case.

4.3.2 Attack 2: TTL Expiry

Our second attack exploits the fact that the AH MAC cannot cover all the fields of the inner IP header. In particular, the TTL and checksum fields are unprotected and so can be manipulated by the attacker. This attack allows us to relax the plaintext requirements in comparison to the previous attack. However, we require that IP packets on the VPN are directed to hosts behind G_B. Again, we begin by assuming that AH replay protection is disabled.

Attack 2, Step 1 We begin with a one-time preparation step. Suppose the attacker captures an arbitrary IPsec-protected packet intended for a host behind G_B. The attacker can manipulate bits in the IV of the CBC-mode ciphertext after the ESP header, with the effect of reducing the TTL field in the inner header to 0. This requires the header checksum to be corrected, and here we rely on the improved method described in Section 4.2.2. For example, supposing the TTL field's original value is `0x40`, then on average 2 trials are needed, while if the original value is `0xFF`, then on average 6.75 trials are needed. Alternatively, we might only assume that the TTL field is uniformly distributed; then, by carefully scheduling the bit flips applied to the TTL and checksum fields in an extension of the method of Section 4.2.2, we can simultaneously reduce the TTL field to 0 and correct the checksum using an expected number of 382 trials.

In each case, after a certain expected number of trials, the attacker succeeds in creating an IPsec-protected packet for which the TTL field in the inner IP header is 0, the checksum in the inner IP header is correct, and the AH MAC on the inner packet verifies. Because the inner packet should be forwarded to a host behind the gateway G_B, such an IP packet should always induce G_B to produce an ICMP response (of type 11 and code 0). We will use this IP packet in step 2 as a carrier packet. In the case when the starting value of the TTL field is not known, we need to be careful to distinguish this ICMP response from any other replies that may arise when the IP header checksum is correct but the TTL has not been successfully set to 0.

Attack 2, Step 2 The attacker now mounts an ESP trailer oracle attack using the carrier packet constructed in step 1, splicing blocks r, c_i^* onto the end of the carrier packet for different values r, starting with the 2^{16} variants in the rightmost 2 bytes of r. As with Attack 1, we rely on ESP's handling of TFC bytes to ensure that the inner packet presented to AH after successful ESP processing at G_B always passes AH's MAC check, even with the blocks r, c_i^* spliced onto the carrier packet. On average, after 2^{15} trials, an ICMP response will be detected in the reverse direction on the VPN between G_A and G_B. This indicates a particular value of r for which the packet ending in r, c_i^* passed the ESP trailer checks. The attack now continues in the usual way.

This attack only applies for encryption algorithms with 128-bit block size, because we must be able to manipulate the TTL field in the inner IP header, and this is located beyond the first 64 bits of the header. In step 2, the attack requires an average of $2^{15} + 14 \cdot 2^7$ trial packet injections to recover any complete 128-bit block of plaintext.

Attack 2 with AH replay protection enabled We can modify the above attack to cope with the situation where AH replay protection is enabled. The main difference is that we can no longer re-use a single carrier packet constructed in a first step, because once AH processing has been triggered (after successful ESP processing), a fixed carrier packet's AH sequence number would always be rejected thereafter. To overcome this, we combine the carrier packet generation and ESP trailer oracle steps. Thus, for each choice of r used in a normal attack, we must splice r, c_i^* onto a sequence of trial packets, with each trial starting with a base packet and attempting to manipulate the TTL field, correct the checksum, pass ESP trailer processing, pass AH processing, and finally generate an ICMP message. Clearly, for each success in this endeavour, the attacker can extract 2 or 1 plaintext bytes (depending on whether the rightmost bytes are being targeted or not), and must move on to a new base packet with a fresh sequence number for each success.

For an assumed inner TTL field of, say `0xFF`, an average of 6.25×2^{16} trials are needed to extract the rightmost 2 bytes of any block, and an average of 6.25×2^{8} trials for each byte thereafter. Extracting each block of plaintext requires the attacker to have gathered 16 IPsec-protected packets with roughly increasing AH sequence numbers, and also to block other traffic on the VPN while the attack is in progress. If nothing is assumed about the starting TTL value, then the attacker would first conduct a reconnaissance phase to ascertain likely TTL values (since only a few possible different values would be expected, depending on the particular OS involved and the number of hops between the end host generating the inner packet and the gateway G_A). This would involve testing possible TTL value and checksum correction masks in a systematic manner in an effort to produce an ICMP response, with an expected number of 382 trials being needed (assuming the TTL field is uniformly distributed). Once the likely TTL values have been determined, the attack can proceed as just described for known TTL values. The attack can still be mounted without a reconnaissance phase, or with unstable inner TTL field values, but it becomes rather expensive in terms of the number of packet injections needed.

4.3.3 Attack 3: Fragmentation

In the previous two attacks we endeavoured not to tamper with authenticated portions of payloads, instead making use of intercepted packets that generate some form of reply at the receiver, or by manipulating portions of the ESP payload that are not protected by AH. Our third attack adopts a different approach, managing to avoid the plaintext requirements of the previous two attacks. We now craft packets that will generate replies whilst completely bypassing AH processing at the receiver. The basic idea is that, after ESP decapsulation of a crafted packet, the receiver discovers that the ESP payload contains only a fragment of the packet that was originally protected by AH; since AH's MAC cannot be verified unless the receiver has the complete packet, the MAC check will not occur and AH will enter a state in which it waits for further fragments. Eventually, this state will time-out, and generate an error message that is detected by the adversary.

Attack 3, Step 1 We begin with a one-time preparation step. Suppose the attacker captures an arbitrary IPsec-protected packet intended for G_B. The attacker can manipulate bits in the ID field and the MF and DF bits by flipping bits in the IV of the CBC-mode ciphertext after the ESP header, with the effect of turning the inner packet (that is still protected by AH) into something that is interpreted by the receiver as a fragment. Here, we need to set the MF bit, possibly unset the DF bit, and then use the ID field to compensate the checksum, as discussed in Section 4.2.2. (Alternatively, we can manipulate the fragment offset and ID fields with similar results.) This can be done even for a block cipher having a 64-bit block, and with a small number of trial masks to determine how to flip bits in the IV. In fact, because of the specific bit flips involved, at most 17 trial packets are needed. This is because for a single bit flip in the IP header, there correspond 17 possible masks, of which only one will result in a valid checksum. The attacker injects all the trial packets in rapid succession, then waits. All the packets will be successfully processed by ESP at G_B, where all but one will have incorrect checksums and be dropped silently by the gateway. The one that has a correct checksum will be interpreted as a fragment, so IPsec will wait for the arrival of further fragments in an attempt to reassemble the original packets *before* any further AH processing takes place at G_B. Eventually, because the further fragments never arrive, the first remaining fragment provokes the production of an ICMP fragment reassembly time exceeded message (of type 11 and code 1) in the reverse direction on the VPN between G_A and G_B, as per Section 3.1. Because of the predictability of the time-out interval, the attacker can correlate the time of appearance of this packet with the time of injection of the trial packets to determine exactly which trial packet was the first one with a correct checksum. This trial packet will be the attacker's carrier packet for the second step in the attack. Note that, whenever this packet is injected into the network towards G_B, it will eventually produce an ICMP response after a suitable time-out period.

Attack 3, Step 2 Now that the preparation phase is complete, the attacker has a carrier packet that can be used to create an ESP trailer oracle. This step works largely as before: the attacker splices blocks r, c_i^* onto the end of the carrier packet for different values r, starting with the 2^{16} variants in the rightmost 2 bytes of r. Here c_i^* is any target block. He injects these 2^{16} trial packets into the network towards G_B, looking for an ICMP message in the reverse direction. He correlates the appearance time of the ICMP message with the injection time of trial packets in order to identify the value of r which led to the ICMP message being produced, again using the predictable nature of the fragmentation time-out. The packet with this value of r must have passed ESP processing, indicating that its trailer field ended with the bytes `00,04`. From this, the rightmost 2 bytes of c_i^* can be deduced in the usual way. The attacker now continues to extract bytes further to the left, again by modifying r, creating trial packets, injecting them and correlating the appearance time of the ICMP message with the injection time of trial packets to identify the successful value of r. Each subsequent plaintext byte that is extracted needs the injection of 2^8 trial packets.

The modifications made to the inner packet in this attack do not cause any problems for AH processing, because the attack bypasses this processing. In this sense, the attack exploits the non-atomic nature of IPsec processing, and the complexities arising from IPsec needing to support IP fragmentation. It works for 64-bit and 128-bit ciphers (using the fact that checksums can be corrected by manipulating the ID field for the 64-bit case). It has no known or chosen plaintext requirements and extracts complete plaintext blocks. Its only disadvantage is that, no matter how fast the attacker can inject the (roughly) 2^{16} trial packets needed, he must wait for the IP fragmentation time-out after each pair of bytes/individual byte. As noted previously, this time-out is recommended to be 60-120 seconds, though it is only 15 seconds in the OpenSolaris implementation. This, then, is the limiting factor for the rate at which the attacker can extract plaintext.

Attack 3 with AH replay protection enabled The key feature of this attack is that AH processing is bypassed altogether. Thus, the carrier packet created in step 1 of the attack continues to produce IP fragmentation time-outs even when used repeatedly in step 2. So, in this case, enabling AH replay protection does not present any additional barrier to the attack. In fact this attack is much easier to mount than our first two attacks when AH replay protection is enabled, because it has no chosen plaintext assumptions, only a single packet is needed in the attack, no control over the traffic flow is needed, and it avoids the complications required to implement the previous attacks without disrupting the traffic flow.

4.4 Attacking Other Configurations

Having given a detailed discussion of three different attack types against the 'AH Transport + ESP Tunnel' configuration, we move on to other configurations in which AH is followed by encryption-only ESP. We omit discussion of 'ESP (auth only) + ESP (enc only)' configurations: since the scope of AH's integrity protection is always greater than that of ESP, it is easy to see that any attack against some 'AH + ESP' configuration will also apply to the corresponding 'ESP (auth only) + ESP (enc only)' configuration.

AH Tunnel + ESP Tunnel: In this configuration the IPsec-protected packets now contain a total of 3 IP headers, since the tunnel processing is applied twice. Here, Attacks 1 and 3 still work with simple modifications, but Attack 2 does not, since the TTL field that needs to be manipulated is the one in the innermost IP header, and this is protected by AH (and cannot be reached from ESP's IV any more).

AH Tunnel + ESP Transport: Here, Attack 2 does not work, since this attack needs to manipulate fields in the inner IP header which can no longer be reached from ESP's IV because of the intervening AH bytes. In Attack 3 we forge an ESP datagram whose payload contains only a fragment of an AH-authenticated IP packet. This is only allowed to happen when ESP is in tunnel mode; in fact there is no way of indicating such an instance when ESP in transport mode is used. As such, Attack 3 cannot be mounted either.

Attack 1 requires some extra assumptions to make it work in this configuration. Firstly, the 'expected' value of the NH byte in the ESP trailer is `51`, indicating AH as the next protocol, rather than `04` as before. However, it may be that more byte values are accepted here by IPsec processing, depending on how liberal the IPsec policies are at the gateway. This increases the success probability when extracting the rightmost 2 bytes, but may leave some uncertainty about the exact value of the rightmost byte of the recovered plaintext block. In practice, only `51` for the NH byte will lead to the production of a response message, since other values will lead to the AH data bytes being misinterpreted as coming from a different upper layer protocol, and the data will most likely not be correctly formatted for that protocol.

Secondly, the attacker relies on ESP processing at G_B to interpret the original data in the ESP trailer and some of the bytes in the spliced blocks r, c_i^* as being TFC padding, and to be able to remove these bytes before submitting the resulting packet to AH processing. For, otherwise, the packet would contain extra bytes and these would cause the AH MAC verification to fail. This requires the ESP implementation at G_B to support TFC padding for transport mode ESP, and to know how to inspect the AH and the inner IP length fields to calculate how many bytes of data should remain after TFC padding has been removed. This places greater expectations on the IPsec implementation, though [62, Section 2.7] states that an IPsec implementation SHOULD be capable of this behaviour.

AH Transport + ESP Transport Here the IPsec-protected packets will only contain a single IP header. This configuration was explicitly ruled out in the previous IPsec architecture [65], and so is not supported by some implementations (e.g. OpenSolaris) but is by others (e.g. Linux). Here Attack 2 fails because there is no inner IP header to manipulate, and Attack 3 does not work, since ESP is in transport mode.

Attack 1 requires some modification in order to work. As with the previous attack, the NH byte in ESP trailer is now expected to have value `51`, and the attacker must rely on ESP processing at G_B to accurately handle TFC padding. This requires the ESP implementation at G_B to support TFC padding for transport mode ESP and the upper layer protocol to include an explicit length field, ruling out the use of TCP in the payload of the carrier packet. This means that we are restricted to using ICMP (or perhaps some kind of UDP packet that always produces a response). Otherwise, the attack works as described previously.

4.5 Experimental Results

Having described how our attacks should operate for an RFC-compliant implementation of the RFCs, we now turn to their experimental validation.

Our experimental set-up is composed of two desktop machines acting as the two stand-alone gateways, a laptop acting as the attacker's platform, and a 10 Mbit Hub. The gateways run OpenSolaris build 134, whereas the attacker's platform runs Linux 2.6. We implemented our attacks in Python 2.6.4 and used the Scapy 2.0 library to intercept and manipulate IP packets and to re-inject them into the network. We decided to use OpenSolaris because it can be configured to perform full padding checks on ESP-protected packets as recommended by [62]. In our experiments, the two gateways were configured to protect their communications using AH in transport mode followed by ESP in tunnel mode, as in the example configuration of Section 4.3. We have not tested our attacks on the other MAC-then-encrypt configurations, but we see no reason why they should not be successful.

We successfully implemented Attacks 1 and 3 on the aforementioned configuration, with the AH replay protection service both disabled and enabled. In OpenSolaris, when keys are set up manually, then replay protection is disabled and there is no way of enabling it (in conformance with [65]). Thus we used manual keying for the scenario where replay protection is disabled, and enabled automated key exchange using IKE in order to turn on the replay protection service. Attack 2 works by generating an ICMP message at the point where the IPsec gateway is about to forward the decrypted packet to the end host in the protected network. As mentioned above, this attack works only when AH is applied in transport mode followed by ESP in tunnel mode. However we discovered that in OpenSolaris, it is not possible to forward packets that are protected by AH in transport mode, preventing us from testing Attack 2 in our experimental set-up. This seems to be a design decision by the OpenSolaris developers: such configurations are perfectly in line with the IPsec RFCs.

All of our attacks rely on the production of ICMP messages, so one might be concerned about the effects of ICMP rate limitation. However, this is not an issue in practice because of the relatively slow speed at which the ICMP packets are produced. In fact the main complication that arises in practice for our attacks is the problem of distinguishing the desired response packets from other IPsec-protected traffic on the VPN. This of course depends on the amount of traffic present on the network. As a first step, if the attacker is able to predict the length of the response packet, then he can filter out all packets whose length does not match this value, and thereby significantly reduce the rate of false positives. If length filtering is not enough or not possible, then one can filter on the basis of 'causation': assuming no network congestion, a response is expected to be seen almost instantly after the packet that caused it was received by the gateway. That is, with sufficiently high probability, the attacker can expect to observe the response within a short time interval of the packet having a correct ESP trailer being sent. The time interval should be short enough that the probability of a false positive appearing within the interval is low. Thus the attacker allows a time interval δ between each attack packet that he sends. Once the attacker has detected what he suspects to be a response packet, he can confirm that this was indeed the case by retesting and checking whether a response is again sent within time δ (this will require the use of a fresh carrier packet whenever replay protection is enabled). This can be repeated multiple times in order to boost the confidence of the detection procedure. Thus in scenarios with high network traffic levels, the attacks may still be realised at the expense of efficiency. Alternatively, if replay protection is disabled or Attack 3 based on fragmentation is used, the attacker can simply capture the packets that are of interest to him and wait for a period of low network traffic in order to carry out his attack.

In our set-up we had minimal spurious network traffic and thus basic filtering based on packet lengths was sufficient. The most computationally intense part of each attack is to extract the rightmost two bytes of the target ciphertext block. Given that we could distinguish a response packet from other traffic accurately enough, we adopted the following strategy in order to speed up the Attack 1: we transmitted all 2^{16} packets at a rate almost equal to the network's capacity. As soon as a response packet was detected, we replayed the last few packets spaced at a greater interval, in order to pinpoint the exact packet which generated the response. We also followed this approach in order to extract the rest of the bytes. On a 10Mbit hub, Attack 1 took on average 70 seconds to recover a 128-bit block of plaintext using a 140-byte carrier packet. It should be noted that if replay protection is enabled, then Attack 1 needs 30 fresh packets to recover a block of plaintext in the manner just described. On the other hand it is possible to sacrifice the attack's time efficiency by transmitting packets at a lower rate such that a packet generating a response can be immediately identified, thereby requiring only 15 fresh packets per 128-bit block of plaintext.

For Attack 3, a similar strategy was adopted. Now the oracle response is only output after the IP fragment reassembly has timed out. In OpenSolaris the default time-out value is 15 seconds. In order to match a response to the packet that generated it, we keep a list of the time instants at which each packet was sent. Then if a response is seen at some time t we search our list for packets that were sent near to the time $t - t_{\text{time-out}}$. This scheme was combined with the method described above where packets are initially sent in a burst, and then two replies are required to accurately locate the packet generating the response. In our experiments, we could locate the packet to lie within a range of roughly 20 packets with the first response, and then replay each packet at intervals of 0.2 seconds and use the second response to locate the desired packet exactly. Following this approach with our experimental set-up Attack 3 recovered a 128-bit block of plaintext in roughly 10 minutes.

4.6 Summary

We have demonstrated attacks against all MAC-then-encrypt configurations of IPsec. These show that such configurations should be avoided in IPsec deployments. We have not found any attacks against encrypt-then-MAC configurations of IPsec.

Our attacks demonstrate the dangers inherent in exposing cryptographic flexibility to users. IPsec in particular places a significant burden on network administrators, requiring them to have sufficient cryptographic expertise in order to select secure configurations. Nothing prevents curious administrators from going "off piste" or protects them from bad advice, such as that to be found in [39, 40, 92] for example. We hope that the attacks given here will illustrate some of the dangers in an accessible form.

Our view of IPsec echoes that expressed in [39]: IPsec, in attempting to be "all things to all men" ends up compromising on security. It would be helpful to standardise

IPsec profiles addressing particular application scenarios rather than allowing a set of components that can be combined and configured in different ways to achieve arbitrary goals. This is because predicting the combined security of distinct cryptographic primitives is quite difficult and requires thorough analysis. While theoretical cryptography has much that is useful to say on this subject [17, 69], it currently falls short of being able to give truly meaningful security guarantees for cryptographic primitives as they are deployed in real protocols. The focus of the next chapters is to address this gap between theory and practice.

Chapter 5

Distinguishable Decryption Failures

Contents

In this chapter we consider security models for symmetric encryption where an adversary can distinguish among distinct decryption failures. Our main purpose is to build models that better reflect the reality of cryptographic implementations, and to surface the security issues that arise from doing so.

5.1 Motivation

Encryption schemes meeting strong notions of security typically introduce redundancy into their ciphertexts, and as a consequence ciphertexts may be deemed invalid during decryption. A scheme's correctness ensures that honestly generated ciphertexts will always decrypt correctly, hence we expect decryption to 'fail' only for ciphertexts that are corrupted during transmission or are adversarially generated. Typically, protocols making use of an encryption scheme report decryption failures to the sender through error messages, and thus the fact that a decryption failure has occurred becomes known to the adversary. After Bleichenbacher's attack on RSA PKCS#1 [21], it became recognised in the academic community that these decryption failures (and the attendant error messages) may leak significant information to an adversary, undermining schemes' confidentiality properties. Other examples in the asymmetric setting were subsequently discovered [52, 71] and called *reaction attacks*. In a very broad sense, Vaudenay's padding oracle attack that we described in Section 3.4 can be seen as an adaptation of Bleichenbacher's attack to the symmetric setting. Vaudenay's ideas were later extended to produce significant attacks against (among others) SSL/TLS [27, 78, 3], IPsec [29, 30], ASP.NET [37], XML encryption [57] and DTLS [2]. The SSH attack by Albrecht, Paterson, and Watson from Section 3.5 also depends in a crucial way on the error messages that are returned by the receiver during decryption.

At a very high level the above-mentioned attacks on symmetric schemes have the common feature that during decryption some information about the plaintext is leaked, due to error messages, their timing, or some other aspect of the implementation. The leaked information is normally quite small, and the power of these attacks really comes from the adversary's ability to amplify this leakage through iteration. That is, given a target ciphertext, an adversary is able to produce a sequence of related ciphertexts which when decrypted will leak more information about the target plaintext. If we now compare this to the IND-CCA security model, it appears that such attacks should be fully accounted for and prevented, given the very conservative approach adopted in this model. Indeed, in the IND-CCA model, the adversary is given full access to a decryption oracle to which it can query any ciphertext except the target ciphertext, and learns either the corresponding plaintext or the fact that decryption fails; and yet this should not leak any information about the target plaintext. Thus if a scheme is IND-CCA-secure, the only information an adversary can gather from a ciphertext is the information leaked during the decryption of that same ciphertext. That is, an adversary is not able to amplify the information leakage through iteration.

The above argument can fail, when information is leaked through decryption failures, as in the attacks in [27, 3, 29, 30, 2]. The decryption algorithm may perform a variety of checks while decrypting a ciphertext, and if any of these checks fail the ciphertext will be deemed invalid. Knowing *why* decryption failed may be more informative to the adversary than the mere fact that decryption failed. Thus attacks that exploit *distinguishable decryption failures* are more powerful than reaction attacks, which only exploit the knowledge of whether a ciphertext is valid or not. While reaction attacks are accounted for in the IND-CCA model, the same cannot be said about distinguishable decryption failure attacks. This is mainly due to the commonly employed formalism where decryption failures always return the same error message ($\perp$), and hence an adversary can learn nothing from invalid ciphertexts in the IND-CCA model. In contrast, in several of the attacks above, an adversary can recover the full plaintext without ever querying a valid ciphertext for decryption, which shows that in practice an adversary can gain information from invalid ciphertexts.

SSL/TLS makes an instructive case study. At a high level, it most commonly uses a Mac-then-Encrypt (MtE) construction, with either a stream cipher or CBC-mode encryption of a block cipher as the encryption scheme. Thus SSL/TLS is covered by Krawczyk's result [69], and one might reasonably conclude that its symmetric encryption scheme is IND-CCA secure. Yet Canvel et al. [27] presented plaintext-recovering attacks against the OpenSSL implementation of SSL/TLS when CBC-mode is used, in which the attacker does nothing other than submit certain ciphertexts for decryption and analyse the results (i.e. the attacker ostensibly operates within the IND-CCA model). The key point, however, is that at the time of Canvel et al.'s attacks in 2003, it was possible to discern whether decryption had failed because the underlying padding needed by CBC-mode was incorrectly formatted or because of a MAC failure. This was possible because they were indicated by different error messages, which even though they were encrypted, were produced at different times during the decryption process. This additional information was sufficient to realise a padding oracle attack, in the style of [93]. Thus, while SSL/TLS may be provably IND-CCA secure in theory, it turned out not to be in practice. Suitable countermeasures involve making it hard for an attacker to learn the cause of decryption failures and were incorporated into the TLS specification from version 1.1 onwards. Meanwhile, building an accurate model of SSL/TLS's symmetric encryption scheme and proving its security has turned out to be a complex task that was only recently completed in [78]. Even there, however, it was necessary to assume that all decryption failures are indistinguishable (since, otherwise, attacks like those of [93, 27, 2, 3] are possible). A similar story could be told for MAC-then-encryption configurations of IPsec, to which the theory in [69] and the attacks of Chapter 4 both apply.

In this chapter we propose to strengthen the existing security definitions for symmetric encryption by letting the adversary distinguish various possible decryption errors. Our main purpose is to build models that better reflect the reality of cryptographic implementations, and to surface the security issues that arise from doing so. We are not the first to make this relaxation (see, for example, [79, 80]), but we are the first to systematically explore its consequences, with some surprising consequences for our understanding of symmetric encryption. Our approach requires the adoption of a slightly different syntax for encryption schemes to the standard one. Now, our decryption algorithm will either return a message from the message space, or an error message from a predetermined finite set of values which we refer to as the *error space*. Technically, then, encryption schemes with multiple errors are a slightly different object from single-error schemes. This approach allows us to handle schemes that can fail in a finite number of distinguishable ways that will be indicated in practice by different error messages. It also enables us to treat attacks in which indistinguishable error messages are returned (perhaps because they are all encrypted, as is the case in SSL/TLS), but in which the errors are returned at a discrete set of times. We note that our approach is equally applicable to the asymmetric setting; here we will restrict our scope to the symmetric setting only.

5.2 The Multiple-Error Setting

A *multiple-error encryption scheme* is defined similarly as in Section 2.3.1 with the following main difference. To a scheme we associate a positive integer n and a set of symbols $\mathcal{S}_\perp = \{\perp_1, \perp_2, \ldots, \perp_n\}$. We refer to this set as the error space of the scheme; it allows the decryption algorithm to indicate invalid ciphertexts with distinct error messages within the error space. The symbol $\perp$ will be used interchangeably to denote a specific error symbol or a variable assuming values from the error space.

Most of the security notions for symmetric encryption that we presented in Chapter 2 apply equally to multiple-error encryption schemes. Note that in our definitions the decryption oracle always returns whatever the decryption algorithm outputs, as opposed to having the experiment return a special symbol if the ciphertext is invalid. Thus the notions that include a decryption oracle are implicitly strengthened by permitting the encryption schemes to have more than one error message.

We now introduce two notions of indistinguishability under ciphertext-validity attack, which can be seen as a strengthened adaption of a similar notion defined by Bauer et al. [7] to the symmetric setting. Here, in addition to an encryption oracle the adversary is given access to a ciphertext-validity oracle which indicates whether a ciphertext is valid or not, and if not, returns the exact error message output by the decryption algorithm.

Definition 5.1: IND-CVA and IND\$-CVA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a multiple-error encryption scheme. For an adversary $\mathcal{A}$ and a bit b, define experiments $\mathbf{Exp}_{\mathcal{SE}}^{\text{ind-cva-b}}(\mathcal{A})$ and $\mathbf{Exp}_{\mathcal{SE}}^{\text{ind\$-cva-b}}(\mathcal{A})$ as shown in Figure 5.1. Both experiments start by calling $\mathcal{K}$ to generate a key K and initialise the states. In the former experiment the adversary $\mathcal{A}$ is given access to a left-or-right encryption oracle $\mathsf{LoR}(\cdot)$, and in the latter it is given a special encryption oracle $\mathsf{Enc\$}(\cdot)$ instead. In both experiments the adversary is additionally given a ciphertext-validity oracle $\mathsf{Val}(\cdot)$. The ciphertext-validity oracle uses $\lightning$ to indicate that the queried ciphertext was valid or has been previously output by the encryption oracle.

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

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

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

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

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

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

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

$\underline{\mathsf{Val}(c)}$
    $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
    if $m \in \mathcal{M}$ $\boxed{\text{if } m \in \mathcal{M} \text{ or } c \in \mathtt{C}}$
        then $m \leftarrow \lightning$
    return $m$
```

Figure 5.1: Experiments to define IND-CVA and IND\$-CVA security. For IND-CVA the boxed code is excluded, whereas for IND\$-CVA the boxed code replaces the code adjacent to it.

It is possible to extend the above two notions to the stateful setting. Interestingly, in the presence of a left-or-right encryption oracle, the $\mathsf{sfVal}(\cdot)$ oracle reduces to a $\mathsf{Val}(\cdot)$ oracle, and therefore IND-sfCVA (defined in the obvious way) would be syntactically equivalent to the IND-CVA experiment. In the case of indistinguishability from random bits, an analogous equivalence is not evident from the syntax. We define this notion below.

Definition 5.2: IND\$-sfCVA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a multiple-error encryption scheme. For an adversary $\mathcal{A}$ and a bit b, define experiment $\mathbf{Exp}^{\text{ind\$-sfcva-b}}_{\mathcal{SE}}(\mathcal{A})$ as shown in Figure 5.2. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is given access to a special encryption oracle $\mathsf{Enc\$}(\cdot)$, and a stateful ciphertext-validity oracle $\mathsf{sfVal}(\cdot)$. The stateful ciphertext-validity oracle returns $\lightning$ until the queries become out-of-sync.

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

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

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

```
$\underline{\mathbf{Exp}_{\mathcal{SE}}^{\text{ind\$-cva-b}}(\mathcal{A})}$
  $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
  $i \leftarrow 0, j \leftarrow 0$
  $\mathtt{C} \leftarrow (), \mathsf{sync} \leftarrow 1$
  $b' \leftarrow \mathcal{A}^{\mathsf{Enc\$}(\cdot), \mathsf{sfVal}(\cdot)}$
  $\mathbf{return}\ b'$

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

$\underline{\mathsf{sfVal}(c)}$
  $j \leftarrow j + 1$
  $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
  $\mathbf{if}\ j > i\ \mathbf{or}\ c \neq \mathtt{C}_j$
      $\mathbf{then}\ \mathsf{sync} \leftarrow 0$
  $\mathbf{if}\ \mathsf{sync} = 1\ \mathbf{or}\ m \in \mathcal{M}$
      $\mathbf{then}\ m \leftarrow \lightning$
  $\mathbf{return}\ m$
```

Figure 5.2: Experiment to define $\mathsf{IND\$\text{-}sfCVA}$ security.

We now turn our attention to integrity of ciphertexts, as defined in Section 2.4 and its stateful variant in Section 2.5. When extending these notions to schemes with multiple errors, some ambiguity arises as to how to interpret the try oracle's functionality. That is, should the try oracle indicate only whether a ciphertext is valid or not, or should it additionally return the exact error message output by the decryption algorithm if the ciphertext is invalid? While the syntax of our definitions from Chapter 2 conforms to the latter interpretation, formulations conforming to the former interpretation are also common in the literature. For single-error schemes the two interpretations are equivalent, but as we shall see in the next section this does not hold in general. For each of the standard and stateful notions we consider both variants and we denote the weaker variant (i.e. the one that is less informative to the adversary) with '$*$'.

Definition 5.3: Ciphertext Integrity in the Multiple-Error Setting. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For an adversary $\mathcal{A}$ define the experiments $\mathbf{Exp}^{\text{int-atk}}_{\mathcal{SE}}(\mathcal{A})$ for $\mathsf{ATK} \in \{\mathsf{CTXT}, \mathsf{CTXT}^*, \mathsf{sfCTXT}, \mathsf{sfCTXT}^*\}$ as shown in Figure 5.3. The experiments start by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$, and either a try oracle $\mathsf{Try}(\cdot)$ or a stateful try oracle $\mathsf{sfTry}(\cdot)$. The output of $\mathsf{Try}(\cdot)$ is suppressed with $\lightning$ in the event that the queried ciphertext is valid or if it has been previously output by the encryption oracle. Similarly, the output of $\mathsf{sfTry}(\cdot)$ is suppressed if the queried ciphertext is valid or in-sync. In the '$*$' variants, error messages are masked with $\perp$, and thus made indistinguishable to the adversary.

The adversary's goal is to make a valid query not previously output by the encryption oracle, or in the stateful case, a valid out-of-sync query. For each experiment, the adversary's advantage is defined as:

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

The scheme $\mathcal{SE}$ is said to be $\mathsf{INT\text{-}ATK}$ secure, if for every adversary $\mathcal{A}$ consuming reasonable resources, its advantage $\mathbf{Adv}^{\text{int-atk}}_{\mathcal{SE}}(\mathcal{A})$ is small.

Although an encryption scheme may have multiple error messages, not all error messages may be 'available' to the adversary. In particular an adversary may not be

```
$\mathbf{Exp}^{\text{int-ctxt}}_{\mathcal{SE}}(\mathcal{A})$ $\boxed{\mathbf{Exp}^{\text{int-ctxt*}}_{\mathcal{SE}}(\mathcal{A})}$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, \mathsf{C} \leftarrow ()$
    $\mathsf{win} \leftarrow 0$
    $\mathcal{A}^{\mathsf{Enc}(\cdot),\mathsf{Try}(\cdot)}$
    return $\mathsf{win}$

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

$\mathsf{Try}(c)$
    $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
    if $c \notin \mathsf{C}$ and $m \notin \mathcal{S}_\perp$
        then $\mathsf{win} \leftarrow 1$
    if $m \notin \mathcal{S}_\perp$ then $m \leftarrow \lightning$
    $\boxed{\text{else } m \leftarrow \perp}$
    return $m$
```

```
$\mathbf{Exp}^{\text{int-sfctxt}}_{\mathcal{SE}}(\mathcal{A})$ $\boxed{\mathbf{Exp}^{\text{int-sfctxt*}}_{\mathcal{SE}}(\mathcal{A})}$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0, j \leftarrow 0, \mathsf{C} \leftarrow ()$
    $\mathsf{sync} \leftarrow 1, \mathsf{win} \leftarrow 0$
    $\mathcal{A}^{\mathsf{Enc}(\cdot),\mathsf{sfTry}(\cdot)}$
    return $\mathsf{win}$

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

$\mathsf{sfTry}(c)$
    $j \leftarrow j + 1, (m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
    if $j > i$ or $c \neq \mathsf{C}_j$
        then $\mathsf{sync} \leftarrow 0$
    if $\mathsf{sync} = 0$ and $m \neq \perp$
        then $\mathsf{win} \leftarrow 1$
    if $m \notin \mathcal{S}_\perp$ then $m \leftarrow \lightning$
    $\boxed{\text{else } m \leftarrow \perp}$
    return $m$
```

Figure 5.3: Experiments to define ciphertext integrity in the multiple-error setting. For INT-CTXT and INT-sfCTXT the boxed code is excluded, whereas for INT-CTXT* and INT-sfCTXT* the boxed code replaces the code adjacent to it.

able to produce (invalid) ciphertexts that generate all possible error messages. We introduce a simple security notion that captures exactly this situation. Informally an encryption scheme is *error-invariant* if no efficient adversary can generate more than one of the possible error messages. Of course any single-error scheme is trivially error invariant.

Definition 5.4: INV-ERR security. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with error space $\mathcal{S}_\perp$. For any $\perp \in \mathcal{S}_\perp$ and an adversary $\mathcal{A}$, define the experiment $\mathbf{Exp}^{\text{inv-err}}_{\mathcal{SE},\perp}(\mathcal{A})$ as shown in Figure 5.4. A key K is first generated by calling $\mathcal{K}$. The adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$ and a decryption oracle $\mathsf{Dec}(\cdot)$.

```
$\mathbf{Exp}^{\text{inv-err}}_{\mathcal{SE},\perp}(\mathcal{A})$
  $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
  $\mathsf{win} \leftarrow 0$
  $\mathcal{A}^{\mathsf{Enc}(\cdot),\mathsf{Dec}(\cdot)}$
  return $\mathsf{win}$

$\mathsf{Enc}(m)$
  $(c, \sigma) \leftarrow \mathcal{E}_K(m, \sigma)$
  return $c$

$\mathsf{Dec}(c)$
  $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
  if $m \in \mathcal{S}_\perp$ and $m \neq \perp$
      then $\mathsf{win} \leftarrow 1$
  return $m$
```

Figure 5.4: INV-ERR experiment for symmetric encryption schemes.

The adversary's goal is to submit a ciphertext to the decryption oracle which results in an error message not equal to $\perp$. The experiment outputs a bit indicating the adversary's success. We define the advantage of an adversary $\mathcal{A}$ with respect to $\perp$ as:

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

The scheme $\mathcal{SE}$ is said to be INV-ERR secure if there exists a unique $\perp \in \mathcal{S}_\perp$ such that for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{inv-err}}_{\mathcal{SE},\perp}(\mathcal{A})$ is small.

Finally we will also consider message authentication schemes that return multiple errors. Similarly this requires replacing the role of the error message $\perp$ with a set of error messages $\mathcal{Q}_\perp = \{\perp_1, \perp_2, \ldots, \perp_n\}$. The UF-CMA and SUF-CMA security notions that we presented in Section 2.2.2 are thus automatically strengthened by permitting the message authentication schemes to return more than one error message. We do not consider the weaker variant (analogous to the $*$ variant for integrity of ciphertexts) where the adversary only returns whether a message-tag pair is valid or not.

5.3 Relations and Separations

5.3.1 Preliminary Note.

An implication from security notion X to security notion Y, indicated by $\mathsf{X} \longrightarrow \mathsf{Y}$, means that any scheme which is X-secure is also Y-secure. More formally there exists a constant $\kappa > 0$ such that for any symmetric encryption scheme $\mathcal{SE}$ and any Y adversary $\mathcal{A}_y$ there exists a X adversary $\mathcal{A}_x$ (consuming similar resources) such that:

$$\mathbf{Adv}^{\mathrm{y}}_{\mathcal{SE}}(\mathcal{A}_y) \leq \kappa \cdot \mathbf{Adv}^{\mathrm{x}}_{\mathcal{SE}}(\mathcal{A}_x)$$

A separation from security notion X to security notion Y indicated by $\mathsf{X} \not\longrightarrow \mathsf{Y}$, means that there exists an encryption scheme which meets notion X but for which we can exhibit an attack showing that it does not meet notion Y. The separation is interesting only if there exists some scheme which meets security notion X, as otherwise the implication $\mathsf{X} \longrightarrow \mathsf{Y}$ is vacuously true. Our separations can be categorised into two types. In the former we will assume that there exists some scheme $\mathcal{SE}$ which meets notion X, and use it to construct a scheme $\overline{\mathcal{SE}}$ which meets notion X but is insecure in the Y sense. From the foregoing discussion, such an assumption is in some sense minimal. In the second type of separations we will assume the existence of pseudorandom functions and UF-CMA MACs to construct a scheme which meets notion X but not notion Y. In this paper for all separations of the latter type we will have that $\mathsf{X} \longrightarrow$ IND-CPA. It is a well-known result that the existence of IND-CPA-secure symmetric encryption implies the existence of pseudorandom functions [56, 53, 48]. In addition a pseudorandom function can be combined with an almost-universal hash function to obtain a variable-input-length pseudorandom function, which in turn yields a UF-CMA MAC. Thus from a theoretical viewpoint the underlying assumptions for either type of separation are equivalent.

Note that when proving a separation we do not require the scheme to have distinct error messages, as we are interested solely in the *existence* of a counterexample showing that the relation under question cannot be established. Secondly any multiple-error scheme which is secure under some notion X implies the existence of a single-error scheme which is also secure under notion X (simply by mapping all error messages to a single error message). Consequently it is best to prove separations using schemes with an error space of *minimal cardinality.* It then follows that the separation also holds for all schemes of higher error-space cardinality.

5.3.2 Straightforward Relations.

In the previous section we explained how the standard security notions can be extended to the multiple-error setting and introduced new ones. Proposition 5.1 depicts some basic relations between these security notions. These relations are self-evident and we state them without proofs.

Proposition 5.1.

$$\begin{tikzcd}
\mathsf{IND\text{-}sfCCA} \arrow[r] & \mathsf{IND\text{-}CCA} \arrow[r] & \mathsf{IND\text{-}CVA} \arrow[r] & \mathsf{IND\text{-}CPA}
\end{tikzcd}$$

$$\begin{tikzcd}
\mathsf{IND\$\text{-}sfCCA} \arrow[r] \arrow[dr] & \mathsf{IND\$\text{-}CCA} \arrow[r] & \mathsf{IND\$\text{-}CVA} \arrow[r] & \mathsf{IND\$\text{-}CPA} \\
 & \mathsf{IND\$\text{-}sfCVA} \arrow[ur] & &
\end{tikzcd}$$

$$\begin{tikzcd}
\mathsf{INT\text{-}sfCTXT} \arrow[r] \arrow[dr] & \mathsf{INT\text{-}CTXT} \arrow[r] & \mathsf{INT\text{-}CTXT}^* \\
 & \mathsf{INT\text{-}sfCTXT}^* \arrow[ur] &
\end{tikzcd}$$

5.3.3 Revisiting Classic Relations.

If a single-error symmetric encryption scheme satisfies both passive confidentiality (IND-CPA) and integrity of ciphertexts (INT-CTXT), then Theorem 2.2 guarantees that it also offers confidentiality against chosen-ciphertext attacks. Theorem 2.3 extends this result to the stateful setting. Often, when analysing a particular scheme, its chosen-plaintext security and ciphertext integrity are proved first, and then these classic results are used to guarantee chosen-ciphertext security. Indeed, the combination of IND-CPA and INT-CTXT (or their stateful versions) has come to be the accepted security notion for authenticated encryption. We proceed to re-examine these relations in the context of multiple-error encryption schemes.

The following theorem serves as the basis for the two separations in Corollaries 5.3 and 5.4, showing that the classic relations no longer hold for multiple-error schemes. We point out that in proving the separations, we adopt the stronger interpretations of ciphertext integrity so as to make the results as strong as possible.

Theorem 5.2: $\mathsf{IND\text{-}CPA} \wedge \mathsf{INT\text{-}sfCTXT} \not\to \mathsf{IND\text{-}CCA}$. *Let $F : \mathcal{K}_e \times \{0,1\}^\ell \to \{0,1\}^n$ be a pseudorandom function, and let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a* UF-CMA *secure MAC with tag length $\ell_{tag} < n$. Consider the stateful symmetric encryption scheme $\mathcal{SE}_1$ having message space $\{0,1\}^{n-\ell_{tag}}$ and error space $\{\perp_0, \perp_1\}$ shown in Figure 5.5. For any* IND-CPA *adversary $\mathcal{A}_{cpa}$ and any* INT-sfCTXT *adversary $\mathcal{A}_{int}$ against $\mathcal{SE}_1$, both making at most $2^\ell - 1$ encryption queries, there exist two corresponding adversaries $\mathcal{A}_{prf}$ and $\mathcal{A}_{uf}$ using roughly the same resources as $\mathcal{A}_{cpa}$ and $\mathcal{A}_{int}$, respectively, such that:*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}_1}(\mathcal{A}_{cpa}) \ \leq 2 \cdot \mathbf{Adv}^{\text{prf}}_{F}(\mathcal{A}_{prf})\,, \tag{5.1a}$$

$$\mathbf{Adv}^{\text{int-sfctxt}}_{\mathcal{SE}_1}(\mathcal{A}_{int}) \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf})\,. \tag{5.1b}$$

Moreover there exist efficient adversaries $\mathcal{A}_{cca}$ and $\mathcal{A}'_{uf}$ such that:

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}_1}(\mathcal{A}_{cca}) \ = 1 - \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}'_{uf})\,. \tag{5.1c}$$

Proof. The correctness of the constructed scheme is easy to verify and we therefore proceed to prove the first part of the theorem. For any adversary $\mathcal{A}_{cpa}$, making at most $2^\ell - 1$ encryption queries, we construct $\mathcal{A}_{prf}$ as follows. Adversary $\mathcal{A}_{prf}$ runs $\mathcal{K}_m$ to get a key for the MAC, it then runs $\mathcal{A}_{cpa}$ and provides it with a simulation of its left-or-right encryption oracle. Essentially $\mathcal{A}_{prf}$ selects a uniformly random bit d and uses its own oracle together with its MAC key to encrypt m_d according to the construction in Figure 5.5, where the pseudorandom function is replaced by its own oracle. Finally, if $\mathcal{A}_{cpa}$'s output is equal to d, then $\mathcal{A}_{prf}$ outputs 1 otherwise it outputs 0. Now when $\mathcal{A}_{prf}$'s oracle is instantiated with F it provides $\mathcal{A}_{prf}$ with a perfect simulation of the IND-CPA experiment. On the other hand when $\mathcal{A}_{prf}$'s oracle is a random function, the ciphertexts returned to $\mathcal{A}_{cpa}$ provide no information

```
Algorithm $\mathcal{K}$
    $K_e \leftarrow_{\$} \mathcal{K}_e$
    $K_m \leftarrow \mathcal{K}_m$
    $\sigma \leftarrow 1,\ \varrho \leftarrow 1$
    $K \leftarrow K_e \parallel K_m$
    return $(K, \sigma, \varrho)$
```

```
Algorithm $\mathcal{E}_K(m, \sigma)$
    $\tau \leftarrow \mathcal{T}_{K_m}(\langle\sigma\rangle_\ell \parallel m)$
    $c \leftarrow F_{K_e}(\langle\sigma\rangle_\ell) \oplus (m \parallel \tau)$
    $\sigma \leftarrow \sigma + 1 \bmod 2^\ell$
    return $(c, \sigma)$
```

```
Algorithm $\mathcal{D}_K(c, \varrho)$
    if $|c| \neq n$ then $\varrho \leftarrow 0$
    if $\varrho = 0$ then
        return $(\perp_0, \varrho)$
    $w \leftarrow F_{K_e}(\langle\varrho\rangle_\ell) \oplus c$
    parse $w$ as $m \parallel \tau$
    $v \leftarrow \mathcal{V}_{K_m}(\langle\varrho\rangle_\ell \parallel m, \tau)$
    if $v =$ valid
        then $\varrho \leftarrow \varrho + 1 \bmod 2^\ell$
    else
        $\varrho \leftarrow 0$
        if $m[1] = 0$ then
            $m \leftarrow \perp_0$
        else $m \leftarrow \perp_1$
    return $(m, \varrho)$
```

Figure 5.5: The scheme $\mathcal{SE}_1$ of Theorem 5.2.

about d, i.e. d is information-theoretically hidden. Therefore we have that:

$$\begin{aligned}
\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}_{prf}) &= \Pr\left[K_e \leftarrow_{\$} \mathcal{K}_e : \mathcal{A}_{prf}^{F_{K_e}(\cdot)} = 1\right] - \Pr\left[f \leftarrow_{\$} \mathsf{Func}(\ell, n) : \mathcal{A}_{prf}^{f(\cdot)} = 1\right] \\
&= \Pr\left[d \leftarrow_{\$} \{0,1\} : \mathbf{Exp}_{\mathcal{SE}_1}^{\mathrm{ind\text{-}cpa\text{-}d}}(\mathcal{A}_{cpa}) = d\right] - \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{SE}_1}^{\mathrm{ind\text{-}cpa}}(\mathcal{A}_{cpa}) - \frac{1}{2} = \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{SE}_1}^{\mathrm{ind\text{-}cpa}}(\mathcal{A}_{cpa}) .
\end{aligned}$$

Inequality (5.1a) thus follows, and we now prove the second inequality.

For any adversary $\mathcal{A}_{int}$ making at most $2^\ell - 1$ encryption queries, adversary $\mathcal{A}_{uf}$ proceeds as follows. It samples a key K_e for the pseudorandom function F and then runs adversary $\mathcal{A}_{int}$. It simulates the encryption oracle by using its own tagging oracle and the pseudorandom function under the sampled key. In addition it also maintains an ordered list of the messages $\mathcal{A}_{int}$ queries to the encryption oracle together with their corresponding ciphertexts. It then simulates the try oracle as follows. As long as $\mathcal{A}_{int}$'s queries are in sync, i.e. they match the ciphertexts in $\mathcal{A}_{uf}$'s list in the exact same order, it returns $\lightning$. Alternatively consider $\mathcal{A}_{int}$'s first out-of-sync query c_i, let this be its i^{th} try query. In this case $\mathcal{A}_{uf}$ first checks that $|c_i| = n$ and if not it halts, otherwise it computes the XOR of c_i and $F_{K_e}(\langle i\rangle_\ell)$. It then parses the result into a message and a tag, prepends the message with the string $\langle i \rangle_\ell$, submits it together with the tag to its verification oracle and halts.

Note that due to the scheme's construction $\mathcal{A}_{int}$ can only win within its first $2^\ell - 1$ try queries. Since we are interested in bounding its advantage we only need to consider the case where $i \leq 2^\ell - 1$. Now $\mathcal{A}_{uf}$ provides $\mathcal{A}_{int}$ with a perfect simulation of the INT-sfCTXT experiment until $\mathcal{A}_{int}$ makes its first out-of-sync try query, at which point $\mathcal{A}_{int}$ will either win or lose the experiment (again due to the scheme's construction). Moreover because $\mathcal{A}_{uf}$'s only verification query corresponds to an out-of-sync query and $\mathcal{A}_{int}$ can only make at most $2^\ell - 1$ encryption queries, it follows that the message prepended with $\langle i \rangle_\ell$ could not have been previously queried by $\mathcal{A}_{uf}$ to its tagging oracle. Thus whenever $\mathcal{A}_{int}$ wins $\mathcal{A}_{uf}$ also wins, and inequality (5.1a) follows.

We conclude the proof by describing adversary $\mathcal{A}_{cca}$ which breaks the IND-CCA security of $\mathcal{SE}_1$. The adversary submits $(0 \,\|\, 0^{n-\ell_{tag}-1}, 1 \,\|\, 0^{n-\ell_{tag}-1})$ to the left-or-right oracle and gets in return a ciphertext c^*. It then submits $c^* \oplus (0^{n-\ell_{tag}-1} \,\|\, 1)$ to the decryption oracle. If the decryption oracle returns $\perp_0$ then $\mathcal{A}_{cca}$ outputs 0, otherwise it outputs 1. Due to the scheme's construction, this adversary will always win except for the case where the decryption oracle returns $m \notin \{\perp_0, \perp_1\}$. However this would imply a MAC forgery. Adversary $\mathcal{A}_{cca}$ can then be easily transformed into a UF-CMA adversary $\mathcal{A}'_{uf}$ against $\mathcal{MA}$ such that equation (5.1c) holds. $\square$

Combining Theorem 5.2 and Proposition 5.1 yields the desired separations.

Corollary 5.3: IND-CPA $\wedge$ INT-CTXT $\not\rightarrow$ IND-CCA. *Let $F : \mathcal{K}_e \times \{0,1\}^\ell \to \{0,1\}^n$ be a pseudorandom function, and let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a* UF-CMA *secure MAC with tag length $\ell_{tag} < n$. Then there exists a symmetric encryption scheme that is both* IND-CPA *secure and* INT-CTXT *secure but that is not secure in the* IND-CCA *sense.*

Corollary 5.4: IND-CPA $\wedge$ INT-sfCTXT $\not\rightarrow$ IND-sfCCA. *Let $F : \mathcal{K}_e \times \{0,1\}^\ell \to \{0,1\}^n$ be a pseudorandom function, and let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a* UF-CMA *secure MAC with tag length $\ell_{tag} < n$. Then there exists a symmetric encryption scheme that is both* IND-CPA *secure and* INT-sfCTXT *secure but that is not secure in the*

IND-sfCCA *sense.*

Note that in proving Theorem 5.2 we resorted to a stateful scheme. Only a stateful scheme can be INT-sfCTXT secure, and therefore the counterexample used to prove Corollary 5.4 needs to be stateful. The same cannot be said however about the separation in Corollary 5.3, and in fact it can be proven more generally using a stateless scheme, but we omit the details for the sake of brevity.

5.3.4 New Relations.

We now go on to investigate how chosen-ciphertext security can be obtained in the multiple-error setting. Given how useful Theorem 2.2 and Theorem 2.3 have turned out to be, it would make sense to attempt to derive analogous relations that hold more generally. The following theorem extends the relation of Theorem 2.2 to schemes with multiple errors.

Theorem 5.5: IND-CVA $\wedge$ INT-CTXT $\longrightarrow$ IND-CCA. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For any* IND-CCA *adversary $\mathcal{A}_{cca}$ there exist adversaries $\mathcal{A}_{cva}$ and $\mathcal{A}_{int}$ consuming similar resources to $\mathcal{A}_{cca}$ such that:*

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(\mathcal{A}_{cca}) \leq \mathbf{Adv}^{\text{ind-cva}}_{\mathcal{SE}}(\mathcal{A}_{cva}) + 2 \cdot \mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}}(\mathcal{A}_{int}) \,. \tag{5.2}$$

Proof. To any IND-CCA adversary $\mathcal{A}_{cca}$ we can associate an IND-CVA adversary $\mathcal{A}_{cva}$ and an INT-CTXT adversary $\mathcal{A}_{int}$. Both $\mathcal{A}_{cva}$ and $\mathcal{A}_{int}$ operate by running $\mathcal{A}_{cca}$, and then attempt to simulate its environment as follows. Adversary $\mathcal{A}_{cva}$ forwards $\mathcal{A}_{cca}$'s left-or-right queries to its own left-or-right oracle, and forwards decryption queries to its validation oracle. If the ciphertext turns out to be invalid it returns the error message to $\mathcal{A}_{cca}$, otherwise it aborts. It then outputs whatever $\mathcal{A}_{cca}$ outputs. Adversary $\mathcal{A}_{int}$ picks a bit uniformly at random, and uses this together with its encryption oracle to simulate $\mathcal{A}_{cca}$'s left-or-right oracle. It forwards decryption queries to its verification oracle and returns any error messages back to $\mathcal{A}_{cca}$.

Let W represent the event $\mathbf{Exp}^{\text{ind-cca-b}}_{\mathcal{SE}}(\mathcal{A}_{cca}) = b$ where b is picked uniformly at random. Let E represent the event that $\mathcal{A}_{cca}$ makes a valid decryption query. We then have that:

$$\begin{aligned}\Pr[\,W\,] &= \Pr\left[\,W \wedge \overline{E}\,\right] + \Pr[\,W \wedge E\,] \\ &\leq \Pr\left[\,W \wedge \overline{E}\,\right] + \Pr[\,E\,]\;.\end{aligned}$$

We now bound each of the terms on the right-hand side of the last inequality. Note that $\mathcal{A}_{int}$ simulates $\mathcal{A}_{cca}$'s environment perfectly until the point where $\mathcal{A}_{cca}$ makes a valid decryption query. Thus it follows that whenever E occurs, $\mathcal{A}_{int}$ wins the INT-CTXT experiment. On the other hand if E does not occur, then $\mathcal{A}_{cva}$'s simulation of $\mathcal{A}_{cca}$'s environment is perfect. Consequently whenever event $W \wedge \overline{E}$ occurs, $\mathcal{A}_{cva}$ wins the IND-CVA experiment. Equation (5.2) follows by combining the above and noting that:

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathcal{SE}}(\mathcal{A}) = 2 \cdot \Pr\left[\,b \leftarrow_{\$} \{0,1\} : \mathbf{Exp}^{\text{ind-atk-b}}_{\mathcal{SE}}(\mathcal{A}) = b\,\right] - 1\,. \qquad (5.3)$$

□

A similar relation can be established for stateful chosen-ciphertext security, and each of these relations can be re-proven for security notions involving indistinguishability from random bits. We state these relations below.

Proposition 5.6.

$$\begin{aligned}\mathsf{IND\text{-}CVA} \;\wedge\; \mathsf{INT\text{-}sfCTXT} &\longrightarrow \mathsf{IND\text{-}sfCCA}\\ \mathsf{IND\$\text{-}CVA} \;\wedge\; \mathsf{INT\text{-}CTXT} &\longrightarrow \mathsf{IND\$\text{-}CCA}\\ \mathsf{IND\$\text{-}sfCVA} \wedge \mathsf{INT\text{-}sfCTXT} &\longrightarrow \mathsf{IND\$\text{-}sfCCA}\end{aligned}$$

5.3.5 Necessity of Strong Ciphertext Integrity.

The above relations can be seen as strengthened variants of Theorems 2.2 and 2.3, where we replaced CPA security with CVA security and adopted the stronger notions of ciphertext integrity. It is natural to ask whether the left-hand side of each relation can be somehow relaxed. We have seen in Corollaries 5.3 and 5.4 that reverting from CVA security to CPA security is not an option. However it is not evident whether it is necessary to require the stronger variants of ciphertext integrity. Theorem 5.7 answers this question by means of a separation, proving that strong ciphertext integrity is necessary for Theorem 5.5 to hold.

Theorem 5.7: IND-CVA $\wedge$ INT-CTXT$^* \not\rightarrow$ IND-CCA. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with a large message space $\mathcal{M}$ and an error space $\{\perp_0\}$, such that it is both* IND-CVA *secure and* INT-CTXT* *secure. Let the length of its ciphertexts be bounded above by 2^ℓ for some integer ℓ. Consider the scheme $\overline{\mathcal{SE}}$ having message space $\mathcal{M}$ and error space $\{\perp_0, \perp_1\}$ shown in Figure 5.6. For any* IND-CVA *adversary $\mathcal{A}_{cva}$ making q_e left-or-right queries, and any* INT-CTXT* *adversary $\mathcal{A}_{int}$ making q_t try queries, there exist adversaries $\mathcal{A}^1_{cva}$, $\mathcal{A}^2_{cva}$, and $\mathcal{A}^1_{int}$ (consuming similar resources to $\mathcal{A}_{cva}$ and $\mathcal{A}_{int}$) such that:*

$$\mathbf{Adv}^{\text{ind-cva}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{cva}) \ \leq \mathbf{Adv}^{\text{ind-cva}}_{\mathcal{SE}}(\mathcal{A}^1_{cva}) + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cva}}_{\mathcal{SE}}(\mathcal{A}^2_{cva}) + \frac{q_e}{|\mathcal{M}|}\,, \tag{5.4a}$$

$$\mathbf{Adv}^{\text{int-ctxt*}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{int}) \leq \mathbf{Adv}^{\text{int-ctxt*}}_{\mathcal{SE}}(\mathcal{A}^1_{int}) + \frac{q_t}{|\mathcal{M}|}\,. \tag{5.4b}$$

Moreover there exists an adversary $\mathcal{A}_{cca}$, making at most $(\ell + \max_{m\in\mathcal{M}}(|m|) + 1)$ decryption queries and one left-or-right query such that:

$$\mathbf{Adv}^{\text{ind-cca}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{cca}) \ = 1\,. \tag{5.4c}$$

The intuition behind the separation is as follows. In the INT-CTXT* experiment the adversary does not get to see the decryption error symbols, whereas in the IND-CVA experiment he does not get access to the plaintext corresponding to valid decryption queries. We therefore introduce a weakness in the scheme that can only be exploited when the adversary has access to both the plaintext and the error symbols from a decryption oracle. The new scheme is defined such that during key generation a message from the message space is chosen at random. Encryption is then redefined such that this particular message always encrypts to the same ciphertext. If an adversary knows this message it would then be able to distinguish the left oracle from the right oracle by querying this message twice. The decryption algorithm is defined so that it leaks the encryption of this message through the

```
Algorithm $\overline{\mathcal{K}}$
  $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
  $m^* \leftarrow_{\$} \mathcal{M}$
  $(c^*, \sigma) \leftarrow \mathcal{E}_K(m^*, \sigma)$
  $(m, \varrho) \leftarrow \mathcal{D}_K(c^*, \varrho)$
  $K_0 \leftarrow (K, m^*, c^*)$
  return $(K_0, \sigma, \varrho)$
```

```
Algorithm $\overline{\mathcal{E}}_{K_0}(m, \sigma)$
  if $(m = m^*)$ then
      $c \leftarrow c^*$
  else
      $(c, \sigma) \leftarrow \mathcal{E}_K(m, \sigma)$
  return $(0 \parallel c, \sigma)$
```

```
Algorithm $\overline{\mathcal{D}}_{K_0}(c, \varrho)$
  parse $c$ as $b \parallel c'$
  if $(b = 0)$ then
      if $(c' = c^*)$ then
          $m \leftarrow m^*$
      else
          $(m, \varrho) \leftarrow \mathcal{D}_K(c', \varrho)$
  else
      $\psi \leftarrow \langle |c^*| \rangle_\ell \parallel c^*$
      if $\langle c' \rangle^{-1} \leq |\psi|$ then
          $d \leftarrow \psi[\langle c' \rangle^{-1}]$
          $m \leftarrow \perp_d$
      else $m \leftarrow \perp_0$
  return $(m, \varrho)$
```

Figure 5.6: The scheme $\overline{\mathcal{SE}}$ of Theorem 5.7.

error symbols. However the adversary needs to decrypt this ciphertext in order to mount the distinguishing attack just described, and hence this vulnerability is only exploitable in the IND-CCA game.

Proof. Correctness of the constructed scheme follows easily from the correctness of the original scheme, and we thus proceed to prove equation (5.4a). Adversary $\mathcal{A}^1_{cva}$ starts by picking a message m^* uniformly at random from the message space and computes ψ by querying (m^*, m^*) to its left-or-right oracle. $\mathcal{A}^1_{cva}$ also submits c^* to its ciphertext-validity oracle to maintain the states of its oracles synchronised and thereby correctly simulate $\overline{\mathcal{SE}}$. It then runs $\mathcal{A}_{cva}$ and simulates its oracles according to the construction in Figure 5.6 using its own oracles. Note that $\mathcal{A}^1_{cva}$ provides a perfect simulation to $\mathcal{A}_{cva}$, unless the latter queries m^*. In that case $\mathcal{A}^1_{cva}$ aborts and outputs a bit chosen uniformly at random.

Let W and W^1 represent respectively the events $\mathbf{Exp}^{\text{ind-cva-b}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{cva}) = b$ and $\mathbf{Exp}^{\text{ind-cva-d}}_{\mathcal{SE}}(\mathcal{A}^1_{cva}) = d$ where b and d are picked uniformly at random. Furthermore let E denote the event that $\mathcal{A}_{cva}$ makes an encryption query which includes

m^*. We then have that:

$$\Pr\left[\,W^1\,\right] = \Pr\left[\,W \wedge \overline{E}\,\right] + \frac{1}{2}\cdot\Pr\left[\,E\,\right]$$

$$\Pr\left[\,W^1\,\right] - \frac{1}{2}\cdot\Pr\left[\,E\,\right] + \Pr\left[\,W \wedge E\,\right] = \Pr\left[\,W \wedge \overline{E}\,\right] + \Pr\left[\,W \wedge E\,\right]$$

$$\Pr\left[\,W^1\,\right] + \frac{1}{2}\cdot\Pr\left[\,E\,\right] \geq \Pr\left[\,W\,\right]$$

$$\mathbf{Adv}^{\text{ind-cva}}_{\mathcal{SE}}(\mathcal{A}^1_{cva}) + \frac{1}{2}\cdot\Pr\left[\,E\,\right] \geq \mathbf{Adv}^{\text{ind-cva}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{cva})\,. \tag{5.5}$$

It now remains to bound $\Pr[\,E\,]$. Note that (due to the details of $\overline{\mathcal{SE}}$'s construction) $\mathcal{A}_{cva}$ can recover the encryption of m^* from its ciphertext-validity oracle, and consequently we cannot bound $\Pr[\,E\,]$ using an information-theoretic argument. Instead we construct adversary $\mathcal{A}^2_{cva}$ such that if $\mathcal{A}_{cva}$ can do significantly better than what is information-theoretically possible, then $\mathcal{A}^2_{cva}$ breaks the IND-CVA security of $\mathcal{SE}$. Adversary $\mathcal{A}^2_{cva}$ proceeds exactly as $\mathcal{A}^1_{cva}$, except that it computes c^* by querying (m^+, m^*) to its left-or-right oracle for some message m^+ chosen uniformly at random. Then if at any point during its runtime $\mathcal{A}_{cva}$ queries m^*, $\mathcal{A}^2_{cva}$ outputs 1 else it outputs 0. It then follows that:

$$\mathbf{Adv}^{\text{ind-cva}}_{\mathcal{SE}}(\mathcal{A}^2_{cva}) = \Pr\left[\,\mathbf{Exp}^{\text{ind-cva-1}}_{\mathcal{SE}}(\mathcal{A}^2_{cva}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind-cva-0}}_{\mathcal{SE}}(\mathcal{A}^2_{cva}) = 1\,\right]$$

$$\geq \Pr\left[\,E\,\right] - \frac{2q_e}{|\mathcal{M}|}\,. \tag{5.6}$$

The second line follows from the fact that by definition $\mathcal{A}^2_{cva}$ outputs 1 exactly when E occurs; whereas in the second experiment $\mathcal{A}^2_{cva}$ has no information about m^* and hence an information theoretic argument can be applied. Combining equations (5.5) and (5.6) yields equation (5.4a).

Adversary $\mathcal{A}^1_{int}$ picks a message m^* uniformly at random, computes c^* using its encryption oracle, and then queries c^* to its try oracle to maintain the states synchronised. It then runs $\mathcal{A}_{int}$ and simulates its environment using its own oracles. Specifically it forwards encryption queries to its own encryption oracle and prepends the resulting ciphertexts with a 0 bit. If $\mathcal{A}_{int}$ queries m^* it returns $0\,\|\,c^*$. As regards try queries, it returns $\perp$ for ciphertexts starting with a 1 bit, and for all other queries it chops off the first bit and forwards the remaining ciphertext to its own try oracle. This provides $\mathcal{A}_{int}$ with a perfect simulation of its environment. Let

Z and Z^1 represent respectively the events that $\mathcal{A}_{int}$ and $\mathcal{A}^1_{int}$ win the $\mathsf{INT\text{-}CTXT}^*$ experiment, and let F represent the event that $\mathcal{A}_{int}$ queries $0 \,\|\, c^*$ to its verification oracle without querying m^* to its encryption oracle. We then have that:

$$\begin{aligned} \Pr[\, Z \,] &= \Pr\left[\, Z \wedge \overline{F} \,\right] + \Pr[\, Z \wedge F \,] \\ &\leq \Pr\left[\, Z^1 \,\right] + \Pr[\, F \,] \\ \mathbf{Adv}^{\text{int-ctxt*}}_{\mathcal{SE}}(\mathcal{A}_{int}) &\leq \mathbf{Adv}^{\text{int-ctxt*}}_{\mathcal{SE}}(\mathcal{A}^1_{int}) + \frac{q_t}{|\mathcal{C}|} \,. \end{aligned} \tag{5.7}$$

The bound on $\Pr[\, F \,]$ follows from the fact that unless $\mathcal{A}_{int}$ queries m^* to its encryption oracle, it has no partial information about c^*. Thus equation (5.4b) follows from equation (5.7) by noting that $|\mathcal{C}|$ is at least as large as $|\mathcal{M}|$ (from the correctness of $\mathcal{SE}$).

We now conclude the proof by describing adversary $\mathcal{A}_{cca}$. Note (from the construction of $\overline{\mathcal{SE}}$) that the decryption of $1 \,\|\, \langle i \rangle_\ell$ leaks the i^{th} bit of the string $\psi = \langle |c^*| \rangle_\ell \,\|\, c^*$ through the returned error message. Thus $\mathcal{A}_{cca}$ starts by making a series of ℓ decryption queries, $1 \,\|\, \langle 0 \rangle_\ell$, $1 \,\|\, \langle 1 \rangle_\ell$, $1 \,\|\, \langle 2 \rangle_\ell$, $\ldots$, $1 \,\|\, \langle \ell - 1 \rangle_\ell$, to recover the value $|c^*|$. It then makes a second series of decryption queries, $1 \,\|\, \langle \ell \rangle_\ell$, $1 \,\|\, \langle \ell + 1 \rangle_\ell$, $\ldots$, $1 \,\|\, \langle \ell - 1 + |c^*| \rangle_\ell$, to recover c^*. It can now recover the message m^* by querying the ciphertext $0 \,\|\, c^*$ to its decryption oracle. Having recovered m^*, it submits the pair (m^*, m°) to its left-or-right oracle, where $m^* \neq m^\circ$. If the returned ciphertext is equal to $0 \,\|\, c^*$ the adversary outputs 0, otherwise it outputs 1. This adversary is always successful, and hence equation (5.4c) follows. □

Note: Theorem 5.7 holds irrespective of whether the scheme $\mathcal{SE}$ is stateless or stateful, and in either case the construction of Figure 5.6 yields a scheme $\overline{\mathcal{SE}}$ that is correspondingly stateless or stateful. This is the main reason for prepending ciphertexts with an extra 0 bit during encryption, and for decrypting ciphertexts starting with a 1 separately. If one is happy to further assume that the scheme $\mathcal{SE}$ is stateless, then a slightly simpler construction is sufficient to prove the separation.

Theorem 5.7 also serves as a separation between $\mathsf{INT\text{-}CTXT}^*$ and $\mathsf{INT\text{-}CTXT}$, showing that the latter is strictly stronger. Separations similar to that of Theorem 5.7 corresponding to the relations of Proposition 5.6 can also be established.

Proposition 5.8.

$$\begin{array}{ccccc}
\mathsf{IND\text{-}CVA} & \wedge & \mathsf{INT\text{-}sfCTXT}^* & \not\rightarrow & \mathsf{IND\text{-}sfCCA} \\
\mathsf{IND\$\text{-}CVA} & \wedge & \mathsf{INT\text{-}CTXT}^* & \not\rightarrow & \mathsf{IND\$\text{-}CCA} \\
\mathsf{IND\$\text{-}sfCVA} & \wedge & \mathsf{INT\text{-}sfCTXT}^* & \not\rightarrow & \mathsf{IND\$\text{-}sfCCA}
\end{array}$$

5.3.6 More Separations

We now present a separation showing that IND-CVA is strictly stronger than IND-CPA. We actually show something slightly stronger, in that the separation also holds for schemes which are error invariant. This separation further serves to point out that, even for single-error schemes, Theorem 5.5 does not reduce to the relation of Bellare and Namprempre from [17].

Theorem 5.9: IND-CPA $\wedge$ INV-ERR $\not\rightarrow$ IND-CVA. *Let $F : \mathcal{K}_e \times \{0,1\}^\ell \to \{0,1\}^n$ be a pseudorandom function, where ℓ is sufficiently large. Then the symmetric encryption scheme $\mathcal{SE}_2$ having message space $\cup_{k\geq 1}\{0,1\}^{nk}$ and error space $\{\perp\}$ shown in Figure 5.7 is such that, for any* IND-CPA *adversary $\mathcal{A}_{cpa}$ making q encryption queries totalling μ bits of plaintext, there exists a corresponding adversary $\mathcal{A}_{prf}$ (consuming similar resources to $\mathcal{A}_{cpa}$) with:*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}_2}(\mathcal{A}_{cpa}) \leq 2 \cdot \mathbf{Adv}^{\text{prf}}_{F}(\mathcal{A}_{prf}) + \left(\frac{\mu}{n} + q\right)\left(\frac{q-1}{2^\ell}\right). \tag{5.8a}$$

Moreover there exists an efficient adversary $\mathcal{A}_{cva}$ such that:

$$\mathbf{Adv}^{\text{ind-cva}}_{\mathcal{SE}_2}(\mathcal{A}_{cva}) = 1\,. \tag{5.8b}$$

Proof. It is easy to verify that the constructed scheme is correct, and since its error space contains only a single element it is trivially INV-ERR. We therefore proceed to prove that it is IND-CPA secure. For any adversary $\mathcal{A}_{cpa}$ we construct adversary $\mathcal{A}_{prf}$ as follows. Adversary $\mathcal{A}_{prf}$ selects a uniformly random bit d, runs $\mathcal{A}_{cpa}$ and

```
Algorithm $\overline{\mathcal{K}}$
    $K \leftarrow_{\$} \mathcal{K}_e$
    $\sigma \leftarrow \varepsilon$, $\varrho \leftarrow \varepsilon$
    return $(K, \sigma, \varrho)$
```

```
Algorithm $\overline{\mathcal{E}}_K(m, \sigma)$
    if $|m| \notin \{\alpha n : \alpha \geq 1\}$
        then return $\perp$
    $p \leftarrow |m|/n$
    parse $m$ as $m_1 \| \ldots \| m_p$
    $m_{p+1} \leftarrow 0^n$, $c_0 \leftarrow_{\$} \{0,1\}^{\ell}$
    for $i \leftarrow 1$ to $p+1$ do
        $c_i \leftarrow F_K(c_0 + i) \oplus m_i$
    $c \leftarrow c_0 \| c_1 \| \ldots \| c_{p+1}$
    return $(c, \sigma)$
```

```
Algorithm $\overline{\mathcal{D}}_K(c, \varrho)$
    if $|c| \notin \{\ell + \alpha n : \alpha \geq 2\}$
        then return $\perp$
    $q \leftarrow (|c| - \ell)/n$
    parse $c$ as $c_0 \| \ldots \| c_q$
    for $i \leftarrow 1$ to $q$ do
        $m_i \leftarrow F_K(c_0 + i) \oplus c_i$
    if $m_q \neq 0^n$ then $m \leftarrow \perp$
    else $m \leftarrow m_1 \| \ldots \| m_{q-1}$
    return $(m, \varrho)$
```

Figure 5.7: The scheme $\mathcal{SE}_2$ of Theorem 5.9.

simulates its left-or-right encryption oracle. It does so by using its own oracle to encrypt m_d according to the construction in Figure 5.7, where the pseudorandom function is replaced by $\mathcal{A}_{prf}$'s oracle. Then if $\mathcal{A}_{cpa}$'s output is equal to d, $\mathcal{A}_{prf}$ outputs 1 otherwise it outputs 0. Note that when $\mathcal{A}_{prf}$'s oracle is instantiated with F it provides $\mathcal{A}_{prf}$ with a perfect simulation of the IND-CPA experiment. On the other hand when $\mathcal{A}_{prf}$'s oracle is a random function, the ciphertexts returned to $\mathcal{A}_{cpa}$ provide no information about d, unless $\mathcal{A}_{prf}$'s oracle is queried on the same input more than once. Let E denote the event that $\mathcal{A}_{prf}$ queries its oracle on the same input more than once when simulating the left-or-right oracle. We then have that:

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}_{prf}) = \Pr\left[\, K \leftarrow_{\$} \mathcal{K}_e : \mathcal{A}_{prf}^{F_K(\cdot)} = 1 \,\right] - \Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(\ell, n) : \mathcal{A}_{prf}^{f(\cdot)} = 1 \,\right]$$

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}_{prf}) + \Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(\ell, n) : \mathcal{A}_{prf}^{f(\cdot)} = 1 \,\right] =$$

$$\Pr\left[\, d \leftarrow_{\$} \{0,1\} : \mathbf{Exp}_{\mathcal{SE}_2}^{\text{ind-cpa-d}}(\mathcal{A}_{cpa}) = d \,\right]$$

bounding the left-hand side and using equation (5.3) on the right-hand side,

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}_{prf}) + \frac{1}{2} \cdot (1 - \Pr[\, E \,]) + \Pr[\, E \,] \geq \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{SE}_2}^{\text{ind-cpa}}(\mathcal{A}_{cpa})$$

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}_{prf}) + \frac{1}{2} \cdot \Pr[\, E \,] \geq \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{SE}_2}^{\text{ind-cpa}}(\mathcal{A}_{cpa}) \,.$$

Furthermore it can be shown that (cf. [12, Lemma 10]):

$$\Pr[\, E \,] \leq \left(\frac{\mu}{n} + q\right)\left(\frac{q-1}{2^{\ell}}\right) .$$

By combining the above we get inequality (5.8a). Now, adversary $\mathcal{A}_{cva}$ proceeds as follows. It queries the message pair $(1^n \parallel 1^n, 1^n \parallel 0^n)$ to the left-or-right oracle, and gets an $\ell + 3n$ bit long ciphertext in return. It then takes this ciphertext, truncates the last n bits, and submits it to the validation oracle. If the oracle returns $\perp$ the adversary outputs 0 (left), else if $\lightning$ is returned it outputs 1 (right). It is easy to see that $\mathcal{A}_{cva}$ always succeeds and therefore its advantage is 1. $\square$

Earlier in this chapter we noted that if the IND-sfCVA experiment is defined in the obvious way, it would be syntactically equivalent to the IND-CVA experiment. In the case of indistinguishability from random bits, an analogous equivalence is not evident from the syntax. Theorem 5.10 settles this in the negative, showing that for indistinguishability from random bits the stateful notion is strictly stronger.

Theorem 5.10: IND\$-CVA $\wedge$ INV-ERR$\not\rightarrow$ IND\$-sfCVA. *Let $F : \mathcal{K}_e \times \{0,1\}^\ell \to \{0,1\}^n$ be a pseudorandom function, where ℓ is sufficiently large. Let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a single-error MAC where $\mathcal{T} : \mathcal{K}_m \times \{0,1\}^* \to \{0,1\}^{\ell_{tag}}$ is pseudorandom. Consider the symmetric encryption scheme $\mathcal{SE}_3$ having message space $\cup_{k\geq 1}\{0,1\}^{nk}$ and error space $\{\perp\}$ shown in Figure 5.8. For any* IND\$-CVA *adversary $\mathcal{A}_{cva}$ making q encryption queries totalling μ bits of plaintext, there exist three adversaries $\mathcal{A}^1_{prf}$, $\mathcal{A}^2_{prf}$, and $\mathcal{A}_{uf}$ with:*

$$\mathbf{Adv}^{\text{ind\$-cva}}_{\mathcal{SE}_3}(\mathcal{A}_{cva}) \leq \mathbf{Adv}^{\text{prf}}_{F}(\mathcal{A}^1_{prf}) + \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}^2_{prf}) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \frac{\mu}{n} \cdot \left(\frac{q-1}{2^\ell}\right) + \frac{q(q-1)}{2^{\ell+n+1}} . \tag{5.9a}$$

Moreover there exist efficient adversaries $\mathcal{A}_{sfcva}$ and $\mathcal{A}'_{uf}$ such that:

$$\mathbf{Adv}^{\text{ind\$-sfcva}}_{\mathcal{SE}_3}(\mathcal{A}_{sfcva}) = 1 - \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}'_{uf}) . \tag{5.9b}$$

Proof. The constructed scheme is similar to that of Theorem 5.9, except that it does not append the message with a block of 0's, and the ciphertext is additionally authenticated with a MAC. As before the scheme is trivially INV-ERR since its error space contains only a single element. We will prove that the scheme is IND\$-CVA secure in two steps. For any adversary $\mathcal{A}_{cva}$ we first construct adversary $\mathcal{A}_{uf}$ and

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

```
Algorithm $\overline{\mathcal{E}}_K(m, \sigma)$
    if $|m| \notin \{\alpha n : \alpha \geq 1\}$
        then return $\perp$
    $p \leftarrow |m|/n$
    parse $m$ as $m_1 \parallel \ldots \parallel m_p$
    $c_0 \leftarrow_\$ \{0,1\}^\ell$
    for $i \leftarrow 1$ to $p$ do
        $c_i \leftarrow F_{K_e}(c_0 + i) \oplus m_i$
    $c \leftarrow c_0 \parallel c_1 \parallel \ldots \parallel c_p$
    $\tau \leftarrow \mathcal{T}_{K_m}(c)$
    return $(c \parallel \tau, \sigma)$
```

```
Algorithm $\overline{\mathcal{D}}_K(\psi, \varrho)$
    if $|\psi| \notin \{\ell + \ell_{tag} + \alpha n : \alpha \geq 1\}$
        then return $(\perp, \varrho)$
    parse $\psi$ as $c \parallel \tau$
    $v \leftarrow \mathcal{V}_{K_m}(c, \tau)$
    if $(v \neq \texttt{valid})$ then
        return $(\perp, \varrho)$
    $q \leftarrow (|c| - \ell)/n$
    parse $c$ as $c_0 \parallel \ldots \parallel c_q$
    for $i \leftarrow 1$ to $q$ do
        $m_i \leftarrow F_{K_e}(c_0 + i) \oplus c_i$
    $m \leftarrow m_1 \parallel \ldots \parallel m_q$
    return $(m, \varrho)$
```

Figure 5.8: The scheme $\mathcal{SE}_3$ of Theorem 5.10.

an IND\$-CPA adversary $\mathcal{A}_{cpa}$ against $\mathcal{SE}_3$. In the second step we then show how to construct adversaries $\mathcal{A}^1_{prf}$ and $\mathcal{A}^2_{prf}$ from any such IND\$-CPA adversary. Combining the two steps yields the desired result.

Adversary $\mathcal{A}_{uf}$ runs $\mathcal{K}_e$ to obtain a key for F, picks a bit uniformly at random, and then runs $\mathcal{A}_{cva}$. It then uses the random bit, the PRF indexed by the generated key, and its own tagging oracle to simulate an Enc\$$(\cdot)$ oracle for $\mathcal{A}_{cva}$ according to the construction of Figure 5.8. It handles validation queries by parsing the queried ciphertext into a 'message' and a tag, and forwards the two to its verification oracle. Adversary $\mathcal{A}_{cpa}$ runs $\mathcal{A}_{cva}$, and simulates its encryption oracle using its own oracle. To all validation queries it responds with $\perp$, and it outputs whatever $\mathcal{A}_{cva}$ outputs. Note that $\mathcal{A}_{cpa}$ also makes q encryption queries totalling μ bits of plaintext. Now let W represent the event that $\mathcal{A}_{cva}$ wins the IND-CVA experiment, and let F represent the event that it makes a successful validation query. It then follows that:

$$\Pr[\,W\,] \leq \Pr\left[\,W \wedge \overline{F}\,\right] + \Pr[\,F\,]\ .$$

We can assume without loss of generality that $\mathcal{A}_{cva}$ never queries to its validation oracle a ciphertext that was previously returned by the encryption oracle. We can then bound each of the terms on the right-hand side of the inequality as follows. First note that $\mathcal{A}_{uf}$ provides $\mathcal{A}_{cva}$ with a perfect simulation of the IND-CVA experiment, and clearly whenever E occurs $\mathcal{A}_{uf}$ successfully forges a tag for a new message.

Contrarily if F does not occur then $\mathcal{A}_{cpa}$ simulates $\mathcal{A}_{cva}$'s environment perfectly, and thus whenever $W \wedge \overline{F}$ occurs, $\mathcal{A}_{cpa}$ wins the IND-CPA experiment. This yields:

$$\mathbf{Adv}^{\text{ind\$-cva}}_{\mathcal{SE}_3}(\mathcal{A}_{cva}) \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_3}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf})\,. \tag{5.10}$$

We now move to the second step of the proof and bound $\mathcal{A}_{cpa}$'s advantage. Towards this aim we define a hybrid experiment $\mathbf{ExpH}$, similar in spirit to the two IND\$-CPA experiments corresponding to each bit value. The hybrid experiment proceeds exactly as the $\mathbf{Exp}^{\text{ind\$-cpa-1}}_{\mathcal{SE}_3}$ experiment except for one detail. In the encryption oracle the intermediate string which constitutes the unauthenticated ciphertext is replaced with a uniformly random string of the same length and the MAC is then applied to this string instead. Thus we have that:

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}_3}(\mathcal{A}_{cpa}) = \left(\Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-1}}_{\mathcal{SE}_3}(\mathcal{A}_{cpa}) = 1\,\right] - \Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{cpa}) = 1\,\right]\right)$$
$$+ \left(\Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{cpa}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-0}}_{\mathcal{SE}_3}(\mathcal{A}_{cpa}) = 1\,\right]\right)\,. \tag{5.11}$$

Now we consider each of the above terms in the braces separately, and in each case consider $\mathcal{A}_{cpa}$'s success in distinguishing between the two experiments. For any adversary $\mathcal{A}_{cpa}$ distinguishing between the two experiments in the first term we can associate a PRF adversary $\mathcal{A}^1_{prf}$ against F. Adversary $\mathcal{A}^1_{prf}$ proceeds by running $\mathcal{K}_m$ to obtain a key for $\mathcal{MA}$, and then runs $\mathcal{A}_{cpa}$. It simulates its encryption oracle by using $\mathcal{MA}$ under the obtained key and its own oracle to recreate the encryption algorithm of Figure 5.8. Then $\mathcal{A}^1_{prf}$ outputs whatever $\mathcal{A}_{cpa}$ outputs. Note that if $\mathcal{A}^1_{prf}$'s oracle is instantiated with F, it perfectly simulates a 'real' encryption oracle for $\mathcal{A}_{cpa}$. On the other hand if its oracle is a random function it simulates the encryption oracle of the hybrid experiment as long as it does not query the random function on the same input more than once. Let E_1 denote the event that $\mathcal{A}^1_{prf}$ queries its oracle on the same input more than once when simulating $\mathcal{A}_{cpa}$'s encryption oracle, let Z_b represent the event that $\mathbf{Exp}^{\text{ind\$-cpa-b}}_{\mathcal{SE}_3}(\mathcal{A}_{cpa}) = 1$, and let Z_H represent the event that $\mathbf{ExpH}(\mathcal{A}_{cpa}) = 1$. The first term on the right hand side of equation (5.11) can then be bounded as follows:

$$\begin{aligned}\Pr[\, Z_1 \,] - \Pr[\, Z_H \,] &\leq \Pr\left[\, Z_1 \mid \overline{E_1} \,\right] - \Pr\left[\, Z_H \mid \overline{E_1} \,\right] + \Pr[\, E_1 \,] \\ &\leq \Pr\left[\, K \leftarrow_{\$} \mathcal{K}_e : \mathcal{A}^1_{prf}{}^{F_K(\cdot)} = 1 \,\right] \\ &\qquad - \Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(\ell, n) : \mathcal{A}^1_{prf}{}^{f(\cdot)} = 1 \,\right] + \Pr[\, E_1 \,] \\ &\leq \mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{A}^1_{prf}) + \frac{\mu}{n} \cdot \left(\frac{q-1}{2^\ell} \right) . \end{aligned} \tag{5.12}$$

The bound on $\Pr[\, E_1 \,]$ follows from Lemma 10 in [12]. Now for any adversary $\mathcal{A}_{cpa}$ distinguishing between the two experiments in the second term we construct a PRF adversary $\mathcal{A}^2_{prf}$ against $\mathcal{T}$. Adversary $\mathcal{A}^2_{prf}$ runs $\mathcal{A}_{cpa}$ and simulates its encryption oracle as follows. It first verifies that queried message is in the message space and outputs $\perp$ otherwise. It then samples a random string of length $\ell + |m|$, where m is the queried message, and submits it to its own oracle. It then appends the oracle's output to the random string and returns it to $\mathcal{A}_{cpa}$. Now when $\mathcal{A}^2_{prf}$'s oracle is instantiated with $\mathcal{T}$, it provides $\mathcal{A}_{cpa}$ with a perfect simulation of the hybrid experiment. Alternatively if its oracle is a random function it simulates the 'random' encryption oracle of the IND\$-CPA experiment, as long as it does not query the same string more than once. Let E_2 denote this event, we then have that:

$$\begin{aligned}\Pr[\, Z_H \,] - \Pr[\, Z_0 \,] &\leq \Pr\left[\, Z_H \mid \overline{E_2} \,\right] - \Pr\left[\, Z_0 \mid \overline{E_2} \,\right] + \Pr[\, E_2 \,] \\ &\leq \Pr\left[\, K \leftarrow_{\$} \mathcal{K}_m : \mathcal{A}^2_{prf}{}^{\mathcal{T}_K(\cdot)} = 1 \,\right] \\ &\qquad - \Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(*, n) : \mathcal{A}^2_{prf}{}^{f(\cdot)} = 1 \,\right] + \Pr[\, E_2 \,] \\ &\leq \mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{A}^2_{prf}) + \frac{q(q-1)}{2^{\ell+n+1}} . \end{aligned} \tag{5.13}$$

The second term on the right-hand-side of the last inequality results from a birthday bound on event E_2. Combining equations (5.10) (5.11) (5.12) (5.13) yields inequality (5.9a). This proves that scheme $\mathcal{SE}_3$ is IND\$-CVA secure. To conclude the proof we now describe an adversary $\mathcal{A}_{sfcva}$ that breaks the IND\$-sfCVA security of this scheme. Adversary $\mathcal{A}_{sfcva}$ queries two distinct messages m_1 and m_2 to its encryption oracle in this exact order, and gets in return two corresponding ciphertexts c_1 and c_2. It then makes an out-of-sync query c_2 to the validation oracle. If the oracle returns $\lightning$ it outputs 1 otherwise it outputs 0. Now if the encryption oracle returned a ‘real’ encryption the validation oracle will always return $\lightning$. Alternatively if c_2 is a random string the probability that the validation oracle returns $\lightning$ is bounded by $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}$ (otherwise there exists a trivial adversary against $\mathcal{MA}$). Inequality (5.9b) thus follows. □

5.4 Multiple-Error Authenticated Encryption

In Section 2.4 we presented the IND-CCA3 notion, put forward by Shrimpton in [90], as a concise and elegant security notion for authenticated encryption. For single-error schemes this notion is equivalent to to the combination of chosen plaintext security and ciphertext integrity. We now present a natural extension of this notion to the multiple error setting in terms of indistinguishability from random bits. Then in Theorem 5.11 we show that this characterisation is equivalent to the combination of chosen-plaintext security, weak chosen ciphertext integrity, and error invariance.

Definition 5.5: IND\$-CCA3 for multiple-error symmetric encryption. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a multiple-error symmetric encryption scheme with error space $\mathcal{S}_\perp$. For an adversary $\mathcal{A}$, an error message $\perp \in \mathcal{S}_\perp$ and a bit b, define experiment $\mathbf{Exp}^{\text{ind\$-cca3-b}}_{\mathcal{SE},\perp}(\mathcal{A})$ as shown in Figure 5.9. First $\mathcal{K}$ is called to generate a key K, an initial encryption state σ, and an initial decryption state ϱ. The adversary $\mathcal{A}$ is then given access to a special encryption oracle $\mathsf{Enc\$}(\cdot)$ and a special decryption oracle $\mathsf{Dec}\varnothing(\cdot)$. When $b = 1$ both oracles behave as normal encryption and decryption oracles. When $b = 0$ then $\mathsf{Enc\$}(\cdot)$ will return a random bit string (of the same length as an actual ciphertext would have been), and $\mathsf{Dec}\varnothing(\cdot)$ will always return $\perp$ (unless the queried ciphertext was output by $\mathsf{Enc\$}(\cdot)$, in which case it will return $\lightning$).

The adversary’s goal is to output a bit b', as its guess of the challenge bit b. The experiment returns b' as well and, for $\perp \in \mathcal{S}_\perp$ and an adversary $\mathcal{A}$, the advantage is defined as:

$$\mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{ind\$-cca3-1}}_{\mathcal{SE},\perp}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind\$-cca3-0}}_{\mathcal{SE},\perp}(\mathcal{A}) = 1\right].$$

The scheme $\mathcal{SE}$ is said to be IND\$-CCA3 secure if there exists $\perp \in \mathcal{S}_\perp$ such that for every adversary $\mathcal{A}$ with reasonable resources its advantage $\mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A})$ is small.

```
$\mathbf{Exp}^{\text{ind\$-cca3-b}}_{\mathcal{SE},\perp}(\mathcal{A})$
    $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
    $i \leftarrow 0$, $\mathtt{C} \leftarrow ()$
    $b' \leftarrow \mathcal{A}^{\mathsf{Enc\$}(\cdot),\mathsf{Dec}\varnothing(\cdot)}$
    return $(b')$

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

$\mathsf{Dec}\varnothing(c)$
    $(m, \varrho) \leftarrow \mathcal{D}_K(c, \varrho)$
    if $b = 0$ then $m \leftarrow \perp$
    if $c \in \mathtt{C}$ then $m \leftarrow \lightning$
    return $m$
```

Figure 5.9: Experiment to define IND\$-CCA3 security for multiple-error schemes.

Theorem 5.11: $\mathsf{IND\$\text{-}CPA} \wedge \mathsf{INT\text{-}CTXT}^* \wedge \mathsf{INV\text{-}ERR} \longleftrightarrow \mathsf{IND\$\text{-}CCA3}$. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with error space $\mathcal{S}_\perp$.*

- *For any $\perp \in \mathcal{S}_\perp$ and any adversary $\mathcal{A}_{cca3}$ there exist adversaries $\mathcal{A}_{cpa}$, $\mathcal{A}_{int}$ and $\mathcal{A}_{err}$ (consuming similar resources to $\mathcal{A}_{cca3}$) such that:*

$$\mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A}_{cca3}) \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{int-ctxt}*}_{\mathcal{SE}}(\mathcal{A}_{int}) + \mathbf{Adv}^{\text{inv-err}}_{\mathcal{SE},\perp}(\mathcal{A}_{err}) . \tag{5.14}$$

- *For any $\perp \in \mathcal{S}_\perp$ and any three adversaries $\mathcal{A}'_{cpa}$, $\mathcal{A}'_{int}$ and $\mathcal{A}'_{err}$ there exist three corresponding adversaries $\mathcal{A}^1_{cca3}$, $\mathcal{A}^2_{cca3}$ and $\mathcal{A}^3_{cca3}$ (consuming similar resources to $\mathcal{A}'_{cpa}$, $\mathcal{A}'_{int}$ and $\mathcal{A}'_{err}$, respectively) such that:*

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}'_{cpa}) \leq \mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A}^1_{cca3}) , \tag{5.15a}$$

$$\mathbf{Adv}^{\text{int-ctxt}*}_{\mathcal{SE}}(\mathcal{A}'_{int}) \leq 2 \cdot \mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A}^2_{cca3}) , \tag{5.15b}$$

$$\mathbf{Adv}^{\text{inv-err}}_{\mathcal{SE},\perp}(\mathcal{A}'_{err}) \leq 2 \cdot \mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A}^3_{cca3}) . \tag{5.15c}$$

Proof. We prove the first part of Theorem 5.11 by showing that for any $\perp \in \mathcal{S}_\perp$

and any IND\$-CCA3 adversary $\mathcal{A}_{cca3}$ we can construct three adversaries $\mathcal{A}_{cpa}$, $\mathcal{A}_{int}$, $\mathcal{A}_{err}$ that correspond to the IND\$-CPA, INT-CTXT*, and INV-ERR experiments respectively. Moreover whenever $\mathcal{A}_{cca3}$ is successful, then at least one of the three constructed adversaries will also be successful. Each of the three adversaries runs $\mathcal{A}_{cca3}$ and attempts to simulate its environment as follows. $\mathcal{A}_{cpa}$ forwards encryption queries to its own $\mathsf{Enc\$}(\cdot)$ oracle and responds to decryption queries always with $\perp$. It then outputs whatever $\mathcal{A}_{cca3}$ outputs. As for $\mathcal{A}_{int}$ and $\mathcal{A}_{err}$, these respond to $\mathcal{A}_{cca3}$'s encryption queries using their own encryption oracle, and hence always returns a valid encryption. Furthermore $\mathcal{A}_{int}$ forwards any decryption queries that $\mathcal{A}_{cca3}$ makes to its $\mathsf{Try}(\cdot)$ oracle, and always returns $\perp$. Finally $\mathcal{A}_{err}$ forwards all decryption queries to its own decryption oracle.

Now let Z_b represent the event that $\mathbf{Exp}^{\text{ind-cca3-b}}_{\mathcal{SE},\perp}(\mathcal{A}_{cca3}) = 1$. For $b = 1$ let E and F denote the respective events where $\mathcal{A}_{cca3}$ queries a ciphertext c to its decryption oracle such that $\mathsf{Dec}\varnothing(c) \in \mathcal{S}_\perp \setminus \{\perp\}$, and $\mathsf{Dec}\varnothing(c) \in \mathcal{M}$. We then have that:

$$\begin{aligned}\Pr[\,Z_1\,] &= \Pr\left[\,Z_1 \wedge \overline{F} \wedge \overline{E}\,\right] + \Pr\left[\,Z_1 \wedge F \wedge \overline{E}\,\right] + \Pr[\,W \wedge E\,] \\ &\le \Pr\left[\,Z_1 \wedge \overline{F} \wedge \overline{E}\,\right] + \Pr\left[\,F \wedge \overline{E}\,\right] + \Pr[\,E\,]\ .\end{aligned}$$

$\mathcal{A}_{cca3}$'s advantage can then be expressed as:

$$\begin{aligned}\mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A}_{cca3}) &= \Pr[\,Z_1\,] - \Pr[\,Z_0\,] \\ &\le (\Pr\left[\,Z_1 \wedge \overline{F} \wedge \overline{E}\,\right] - \Pr[\,Z_0\,]) + \Pr\left[\,F \wedge \overline{E}\,\right] + \Pr[\,E\,]\ .\end{aligned}$$

Now each of the three terms on the right-hand side of the last inequality can be bounded as follows. Note that $\mathcal{A}_{err}$ provides $\mathcal{A}_{cca3}$ with a perfect simulation of the IND\$-CCA3 experiment for the case when $b = 1$. Thus whenever E occurs, $\mathcal{A}_{err}$ wins the INV-ERR experiment for $\perp$. On the other hand if E does not occur then $\mathcal{A}_{int}$ provides $\mathcal{A}_{cca3}$'s with a perfect simulation of the IND\$-CCA3 experiment for the case when $b = 1$. This is true until F occurs, at which point $\mathcal{A}_{int}$ wins the INT-CTXT* experiment. Finally if E and F do not occur, then $\mathcal{A}_{cpa}$ provides $\mathcal{A}_{cca3}$ with a perfect simulation of IND\$-CCA3 experiment. It then follows that the first term corresponds to $\mathcal{A}_{cpa}$'s advantage. Combining the above yields inequality (5.14). Note that each of the three adversaries uses similar resources as $\mathcal{A}_{cca3}$.

The second part of the theorem is easier to prove. Adversary $\mathcal{A}^1_{cca3}$ runs $\mathcal{A}'_{cpa}$, forwards encryption queries to its own $\mathsf{Enc\$}(\cdot)$ oracle and outputs whatever $\mathcal{A}'_{cpa}$ outputs. Since this provides $\mathcal{A}'_{cpa}$ with a perfect simulation of its environment, it follows that they both have the same advantage. Adversary $\mathcal{A}^2_{cca3}$ runs $\mathcal{A}'_{int}$ and simulates its oracles using its $\mathsf{Enc\$}(\cdot)$ oracle and its $\mathsf{Dec\varnothing}(\cdot)$ oracle. If at any point $\mathcal{A}'_{int}$ queries a ciphertext (not previously returned by the encryption oracle) which decrypts successfully, then $\mathcal{A}^2_{cca3}$ halts and outputs 1. Otherwise it outputs a uniformly-random bit. Note that when $b = 1$ $\mathcal{A}^2_{cca3}$ provides $\mathcal{A}'_{int}$ with a perfect simulation of its environment, but when $b = 0$ $\mathcal{A}'_{int}$ has zero probability of winning. Inequality (5.15b) then follows from:

$$\begin{aligned}\mathbf{Adv}^{\text{ind\$-cca3}}_{\mathcal{SE},\perp}(\mathcal{A}^2_{cca3}) &= \mathbf{Exp}^{\text{ind\$-cca3-1}}_{\mathcal{SE},\perp}(\mathcal{A}^2_{cca3}) - \mathbf{Exp}^{\text{ind\$-cca3-0}}_{\mathcal{SE},\perp}(\mathcal{A}^2_{cca3}) \\ &= \frac{1}{2} \cdot (1 - \mathbf{Adv}^{\text{int-ctxt}*}_{\mathcal{SE}}(\mathcal{A}'_{int})) + \mathbf{Adv}^{\text{int-ctxt}*}_{\mathcal{SE}}(\mathcal{A}'_{int}) - \frac{1}{2} \\ &= \frac{1}{2} \cdot \mathbf{Adv}^{\text{int-ctxt}*}_{\mathcal{SE}}(\mathcal{A}'_{int}))\,.\end{aligned}$$

Adversary $\mathcal{A}^3_{cca3}$ proceeds in a similar fashion. It runs $\mathcal{A}'_{err}$ and simulates its oracles using its $\mathsf{Enc\$}(\cdot)$ oracle and its $\mathsf{Dec\varnothing}(\cdot)$ oracle. If at any point $\mathcal{A}'_{int}$ queries a ciphertext which returns an error symbol in $\mathcal{S}_\perp \setminus \{\perp\}$, then $\mathcal{A}^3_{cca3}$ halts and outputs 1. Otherwise it outputs a uniformly-random bit. Again when $b = 1$ $\mathcal{A}^3_{cca3}$ provides $\mathcal{A}'_{err}$ with a perfect simulation of its environment, but when $b = 0$ $\mathcal{A}'_{err}$ has zero probability of winning. Inequality (5.15c) then follows as in the previous case. Finally note that in all three cases the respective constructed IND\$-CCA3 adversaries use the same resources as $\mathcal{A}'_{cpa}$, $\mathcal{A}'_{int}$ and $\mathcal{A}'_{err}$. □

It can be similarly shown that:

Proposition 5.12.

$$\mathsf{IND\text{-}CPA} \wedge \mathsf{INT\text{-}CTXT}^* \wedge \mathsf{INV\text{-}ERR} \longleftrightarrow \mathsf{IND\text{-}CCA3}\,.$$

It is easy to see that IND\$-CCA3 security guarantees IND\$-CCA security in the multiple error setting. In fact we can say something slightly stronger, as indicated in Proposition 5.13. The proof is straightforward and we omit it.

Proposition 5.13.

$$\mathsf{IND\$\text{-}CCA3} \longrightarrow \mathsf{IND\$\text{-}CVA} \wedge \mathsf{INT\text{-}CTXT} \longrightarrow \mathsf{IND\$\text{-}CCA}$$

$$\mathsf{IND\text{-}CCA3} \longrightarrow \mathsf{IND\text{-}CVA} \wedge \mathsf{INT\text{-}CTXT} \longrightarrow \mathsf{IND\text{-}CCA}$$

5.5 The Security of Encode-then-Encrypt-then-MAC

Results of Bellare and Namprempre [17] and Krawczyk [69] provide formal evidence for preferring Encrypt-then-MAC (EtM) over other generic compositions like MAC-then-encrypt (MtE). On the other hand, by combining results from [69] and [13], it can be shown that MtE is actually IND-CCA secure when instantiated with CBC-mode encryption or a secure stream cipher (instantiated using counter-mode encryption, for example). Thus the analysis of [17, 69] does not help to separate EtM and MtE when both are suitably instantiated.

Nonetheless practical secure communications systems (employing CBC and counter-mode encryption) based on EtM have so far proved themselves less vulnerable to attack than ones based on MtE. For example, attacks on TLS in [27, 2, 3] and the IPsec attacks from Chapter 4 exploit weaknesses in specific MtE constructions, while attacks against deployed EtM constructions seem rarer.

Reconsidering the EtM and MtE compositions in the multiple-error setting provides new formal grounds for preferring the EtM composition. In what follows, we show that the EtM composition enjoys a robust form of security (in a sense to be made precise). We then go on to show how the above-mentioned attacks on specific MtE constructions can be captured in our multiple-error setting.

To make our considerations more realistic, in place of EtM, we actually consider an encode-then-encrypt-then-MAC (EEM) composition, where the encoding step accounts for the pre-processing (such as padding) that is common in practical schemes. Similarly, we will consider the MAC-then-Encode-then-Encrypt (MEE) composition in place of MtE when discussing attacks.

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

```
Algorithm $\overline{\mathcal{E}}_K(m, \sigma)$
  $u \leftarrow \mathcal{EC}(m)$
  $(c, \sigma) \leftarrow \mathcal{E}_{K_e}(u, \sigma)$
  $\tau \leftarrow \mathcal{T}_{K_m}(c)$
  return $(c \parallel \tau, \sigma)$
```

```
Algorithm $\overline{\mathcal{D}}_K(\psi, \varrho)$
  if $|\psi| < \ell_{tag} + 1$ then
      return $(\perp_0, \varrho)$
  parse $\psi$ as $c \parallel \tau$
  $v \leftarrow \mathcal{V}_{K_m}(c, \tau)$
  if $v \in \mathcal{Q}_\perp$ then
      return $(v, \varrho)$
  $(u, \varrho) \leftarrow \mathcal{D}_{K_e}(c, \varrho)$
  if $u \in \mathcal{S}_\perp$ then
      return $(u, \varrho)$
  $m \leftarrow \mathcal{DC}(u)$
  return $(m, \varrho)$
```

Figure 5.10: The generic Encode-then-Encrypt-then-MAC composition $\mathcal{EEM}$ with distinguishable decryption failures.

Encoding Schemes. Formally an encoding scheme[1] $\mathcal{ES} = (\mathcal{EC}, \mathcal{DC})$ is a pair of algorithms with an associated word space $\mathcal{W} \subseteq \{0,1\}^*$ and error space $\mathcal{U}_\perp$. The *encoding* algorithm $\mathcal{EC}$, which may be probabilistic, takes as input a word $w \in \mathcal{W}$ to return a codeword $u \in \{0,1\}^*$. The deterministic *decoding* algorithm $\mathcal{DC}$ takes as input a codeword $u \in \{0,1\}^*$ to return a word in $\mathcal{W}$ or an error message in $\mathcal{U}_\perp$. We require that for all $w \in \mathcal{W}$ it hold (with probability 1) that $w = \mathcal{DC}(\mathcal{EC}(w))$. Furthermore, an encoding scheme is said to be *length-regular* if for all $w_1, w_2 \in \mathcal{W}$ such that $|w_1| = |w_2|$ it holds (with probability 1) that $|\mathcal{EC}(w_1)| = |\mathcal{EC}(w_2)|$.

Our EEM composition is specified in Figure 5.10. Theorem 5.14 shows that the EEM composition is robust, in the sense that it provides IND-CVA and INT-CTXT security, and therefore IND-CCA security, in the multiple-error setting. The result holds irrespective of the encoding scheme used (and any error messages it returns) and independent of whatever error messages the encryption component returns, so long as the encryption component is IND-CPA and the MAC is SUF-CMA.

Theorem 5.14: EEM provides IND-CVA + INT-CTXT. *Suppose $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ is a symmetric encryption scheme with message space $\mathcal{M}$ and error space $\mathcal{S}_\perp$. Let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a MAC with error space $\mathcal{Q}_\perp$ producing tags of length ℓ_{tag}. Let $\mathcal{ES} = (\mathcal{EC}, \mathcal{DC})$ be a length-regular encoding scheme with word space*

[1]Note that in the next chapter we will define encoding schemes differently.

$\overline{\mathcal{M}}$, error space $\mathcal{U}_\perp$, and whose range is contained in $\mathcal{M}$. Figure 5.10 then defines a symmetric encryption scheme $\mathcal{EEM}$ with message space $\overline{\mathcal{M}}$ and error space $\overline{\mathcal{S}_\perp} = \mathcal{S}_\perp \cup \mathcal{Q}_\perp \cup \mathcal{U}_\perp \cup \{\perp_0\}$, for some $\perp_0 \notin \mathcal{S}_\perp \cup \mathcal{Q}_\perp \cup \mathcal{U}_\perp$. For any IND-CVA *adversary $\mathcal{A}_{cva}$ and any* INT-CTXT *adversary $\mathcal{A}_{int}$ against $\mathcal{EEM}$, there exist adversaries $\mathcal{A}_{cpa}$, $\mathcal{A}^1_{suf}$, and $\mathcal{A}^2_{suf}$ such that:*

$$\mathbf{Adv}^{\text{ind-cva}}_{\mathcal{EEM}}(\mathcal{A}_{cva}) \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{suf-cma}}_{\mathcal{MA}}(\mathcal{A}^1_{suf}) \,, \tag{5.16}$$

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{EEM}}(\mathcal{A}_{int}) \leq \mathbf{Adv}^{\text{suf-cma}}_{\mathcal{MA}}(\mathcal{A}^2_{suf}) \,. \tag{5.17}$$

Moreover, these adversaries consume similar resources to $\mathcal{A}_{cva}$ and $\mathcal{A}_{int}$.

Proof. Correctness of the constructed scheme follows easily from the correctness of its constituent schemes, and we thus proceed to prove its security. We start by proving inequality (5.16).

Adversary $\mathcal{A}_{cpa}$ simply runs $\mathcal{K}_m$ to get a key for the MAC and then runs $\mathcal{A}_{cva}$. It answers its left-or-right encryption queries by first encoding both messages, it then submits them to its own oracle, computes a tag for the resulting ciphertext and returns the ciphertext concatenated with the tag. Validation queries are handled by extracting the tag from the submitted ciphertext, verifying the tag on the remaining string using the derived MAC key, and returning the output to $\mathcal{A}_{cva}$. If the submitted ciphertext is shorter than ℓ_{tag} it returns $\perp_0$ instead. It then outputs whatever $\mathcal{A}_{cva}$ outputs.

Adversary $\mathcal{A}^1_{suf}$ runs $\mathcal{K}_e$ to get an encryption key, picks a bit uniformly at random, and uses these together with its tagging oracle to simulate $\mathcal{A}_{cva}$'s left-or-right oracle. It handles decryption queries by extracting the tag from the submitted ciphertext, and submitting the tag together with the remaining string to its verification oracle, and forwards the output to $\mathcal{A}_{cva}$. If the submitted ciphertext is shorter than ℓ_{tag} it returns $\perp_0$ instead.

Now let W represent the event $\mathbf{Exp}^{\text{ind-cva-b}}_{\mathcal{EEM}}(\mathcal{A}_{cva}) = b$ where b is picked uniformly at random. Let E represent the event that $\mathcal{A}_{cva}$ makes a validation query which returns an error message in $\overline{\mathcal{S}_\perp} \setminus \mathcal{Q}_\perp$. We then have that:

$$\begin{aligned}\Pr[\,W\,] &= \Pr\left[\,W \wedge \overline{E}\,\right] + \Pr[\,W \wedge E\,] \\ &\leq \Pr\left[\,W \wedge \overline{E}\,\right] + \Pr[\,E\,] \,.\end{aligned}$$

We bound each term on the right-hand side of the last inequality as follows. Note that $\mathcal{A}^1_{suf}$ simulates $\mathcal{A}_{cva}$'s environment perfectly until one of $\mathcal{A}_{cva}$'s queries results in a forgery for $\mathcal{A}^1_{suf}$. It then follows that whenever E occurs, $\mathcal{A}^1_{suf}$ wins the SUF-CMA experiment. On the other hand if E does not occur, then $\mathcal{A}_{cpa}$'s simulation of $\mathcal{A}_{cva}$'s environment is perfect. Consequently whenever event $W \wedge \overline{E}$ occurs, $\mathcal{A}_{cpa}$ wins the IND-CPA experiment. Equation (5.16) then follows by combining the above and using equation (5.3).

Adversary $\mathcal{A}^2_{suf}$ runs $\mathcal{K}_e$ to get an encryption key, picks a bit uniformly at random, and uses these together with its tagging oracle to simulate $\mathcal{A}_{int}$'s encryption oracle. For each encryption query that $\mathcal{A}_{int}$'s submits, it first encodes the message and then encrypts it with $\mathcal{E}_K$. It then obtains a tag for the resulting ciphertext from its own oracle, and returns the ciphertext concatenated with the tag. Queries to the Try$(\cdot)$ oracle are handled by extracting a tag from the ciphertext, and submitting tag together with the remaining string to its verification oracle, and the output is returned to $\mathcal{A}_{int}$. On the other hand if the submitted ciphertext cannot be parsed $\perp_0$ is returned. Note that $\mathcal{A}^2_{suf}$ provides $\mathcal{A}_{int}$ with a perfect simulation of the INT-CTXT experiment until the point at which $\mathcal{A}_{int}$ makes a successful try query. Moreover whenever $\mathcal{A}_{int}$ forges a ciphertext, $\mathcal{A}^2_{suf}$'s corresponding verification query will also constitute a forgery. Inequality (5.17) thus follows. □

As a complement to the above result, it is instructive to model attacks on instantiations of the MAC-then-Encode-then-Encrypt (MEE) composition in our multiple-error setting.

- TLS uses a MEE composition in which the encoding step involves the addition of padding having a specific format. This format should be checked for upon decryption, with a failure resulting in an error message. Likewise, the MAC verification may fail, resulting in an error message. Error messages in TLS are encrypted in general, and MAC failures and padding failures are indicated by the same error message. The attacks on TLS [27] and on DTLS [2] use timing differences to distinguish MAC failures from padding failures. These differences can be modelled by introducing distinct error messages for the two failure events (even if at the byte level, the messages are indistinguishable).

- Certain configurations of IPsec use a MEE composition to cryptographically protect IP packets. The security of these configurations were studied in detail in Chapter 4. Here, the encoding step includes a padding portion as well as a header portion, and it is the ability to discern between malformed padding and a malformed header that gives rise to the attacks in Chapter 4. In fact, malformed padding leads to packets being silently dropped, while malformed headers lead to encrypted error messages being sent on the network. Again, the attacks can be modelled by introducing distinct error messages for the different events, even though one of the events does not result in an actual error message being sent (since the absence of a message also leaks information to the adversary).

- The recent Lucky 13 attack on TLS [3] exploits timing differences arising in HMAC's verification algorithm. More specifically each compression function evaluation in HMAC results in additional processing time during decryption that can be detected by the adversary from the time delay in returning TLS's MAC failure message; the size of the delay relates to the amount of TLS padding previously removed and can be used to infer plaintext in an extension of Vaudenay's padding oracle attack [93]. This timing channel can be modelled in our framework by transforming HMAC into a multiple-error MAC. Then the error messages that this version of HMAC returns can be easily predicted from the length of the string on which the tag is to be verified. It follows from this observation that any proof of SUF-CMA security for the usual single-error HMAC can be extended to this multiple-error version of HMAC. So, while this multiple-error HMAC is still SUF-CMA secure, its interaction with the TLS padding renders the MEE composition used in TLS insecure. By contrast, as established in Theorem 5.14, an EEM composition would not be compromised by such an implementation flaw.

Finally, we point out that it is also possible to prove that EEM provides IND-CCA3 security if its MAC component only has a single error message. We omit the details.

5.6 Summary

Our work can be seen to fall in line with other approaches in cryptography, such as nonce-based symmetric encryption [85], where one aims to design cryptographic schemes whose security is more resilient to flaws introduced during implementation. Nonce-based encryption schemes only require a nonce as their auxiliary input (except for the secret key), which need be neither random, nor secret, nor unpredictable; in fact it can even be adversarially generated (as long as no value is ever repeated). Thus nonce-based schemes alleviate the security requirements for generating and handling the auxiliary input in the scheme's implementation and are therefore less prone to implementation vulnerabilities. Similarly our work can be seen as relaxing the assumption that decryption failures are indistinguishable to the adversary, an assumption that is implicit in almost every treatment of symmetric encryption found in the literature. By adopting a syntax where distinct decryption failure return distinct error messages, and proving security in this framework, one obtains schemes whose security is more resilient to implementation flaws.

Chapter 6

Ciphertext Fragmentation

Contents

In this chapter we study symmetric encryption in the presence of ciphertext fragmentation. We formalise this setting and define notions for confidentiality, boundary hiding, and denial of service in the presence of ciphertext fragmentation. The chapter concludes with constructions of encryption schemes that meet these notions.

6.1 Introduction

The work in this chapter is motivated by the SSH attack of Albrecht, Paterson and Watson, described in Section 3.5, and the IPsec attacks by Degabriele and Paterson, described in Chapter 4. What is common to these attacks is that they exploit the data fragmentation mechanisms in each of the corresponding protocols. Data sent over networks is often fragmented, meaning that it is broken up into smaller pieces, or packets. If the data is encrypted, the receiver first has to determine what constitutes a complete ciphertext in order to decrypt it and obtain the underlying message. An exception to this, is when on-the-fly decryption is required, but this is known to reduce security [58]. Reconstruction of the original ciphertext by the receiver can be accomplished by various methods. As we saw in earlier chapters, SSH uses an encrypted length field that tells the receiver how many bytes are needed before the complete ciphertext has arrived. IPsec on the other hand, employs the system of Identification field, More Fragments flag, and Fragment Offset field that it inherits from IP. The SSH and IPsec attacks highlight the conflicts that can arise between confidentiality and support for data fragmentation.

With the exception of [79], ciphertext fragmentation has been ignored by the theoretical community. Security models tend to abstract out data fragmentation, possibly on the assumption that it is immaterial to the security of encryption schemes. Furthermore, ciphertext fragmentation is not only detrimental to confidentiality, but also to more practical security goals, such as traffic analysis. In fact the SSH designers' decision to encrypt the length field (as opposed to the case of TLS where the length field is in the clear) appears to be intended to mitigate traffic analysis, since a cleartext length field allows the delineation of ciphertext boundaries, thereby immediately revealing the ciphertext lengths. Moreover the mechanisms required to support ciphertext fragmentation may introduce new types of security vulnerabilities. Consider SSH for instance, there an adversary is able to exploit the bit-flipping property of the CBC and CTR modes (see Section 2.3.3) to alter the length field that occupies the first 32 bits of plaintext. If the length is maliciously increased to a very large value (say, $2^{32} - 1$, the maximum possible value for a 32-bit field), then the receiver will continue listening for ciphertext fragments awaiting message completion until 2^{32} bytes of data have been received. Only then will SSH's MAC verification be conducted and the message rejected. The application (or user) receiving data from the SSH connection experiences this as an SSH connection hang, a form of Denial-of-Service. As explained in Section 3.2, OpenSSH limits the value of the length field to 2^{18} so as to mitigate against such DoS attacks. However this comes at the expense of limiting the maximum message length.

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

To this end, we initiate a general study of the security of symmetric encryption schemes against fragmentation attacks, not only in terms of message confidentiality, but also in terms of length-hiding and prevention of fragmentation-enabled Denial-of-Service (DoS) attacks against the receiver. To the best of our knowledge, the latter two goals for encryption, i.e. length-hiding (or, more precisely, hiding ciphertext boundaries in a ciphertext stream) and DoS prevention have not been previously studied, partly because the corresponding threats are not present if encryption is treated as being atomic. The adversarial capabilities we define are general enough to model a wide class of fragmentation attacks, including but not limited to the ones from [1, 30].

We complement our new security definitions with efficient cryptographic constructions based on standard primitives meeting the new goals. While it may be relatively easy to achieve each security goal independently, it transpires that it is not straightforward to achieve two or three of the aforementioned goals simultaneously and one of our schemes is the first to do so.

6.1.1 Related Work

Our fragmented approach bears more than a passing resemblance to work on on-line encryption [11, 5, 6, 24, 41, 42, 43, 58]. However, whereas the on-line setting concerns a single continuous message and ciphertext, with each block of plaintext leading to a block of ciphertext being output during encryption (and vice-versa during decryption), our setting concerns atomic encryption (reflecting how many secure protocols operate) but allows fragmented decryption of ciphertexts. Moreover, we extensively treat the case of active adversaries, a topic that has not achieved much attention in the on-line literature, and we consider more than just confidentiality security notions.

Paterson and Watson [79] presented a formal analysis of SSH in counter mode, as implemented in OpenSSH, in a security model that captures ciphertext fragmentation. This obviously bears some similarity to our work, but there are some important differences in scope as well as in the approach that we adopt. While the aim of [79] is to prove secure the OpenSSH implementation of SSH-CTR in a more realistic security model, our scope is to study encryption schemes supporting ciphertext fragmentation and their security more generally. In the context of our scope, the work of Paterson and Watson suffers from the following limitations. Firstly they do not consider boundary-hiding and denial-of-service security. Secondly, a correctness requirement is completely missing. As we shall see this is non-trivial to define. Thirdly, their confidentiality definition [79, Definition 2], which is also based on Bellare et al.'s IND-sfCCA notion, is tailored specifically to work with SSH. In particular, the security experiments refer *directly* to quantities that are SSH specific. For instance the length field, sequence number and buffer *as used by SSH* are also crucial for the definition of security in [79]. Furthermore, a stateful-failing behaviour, where if one decryption call fails all subsequent decryption calls will also fail, is incorporated in the security experiment in [79]. We believe it should be up to the scheme whether to behave in such a way, enforcing it as part of the security experiment results in a weaker security notion. Overall we find our definition to be simpler, cleaner, and at the same time more general.

6.2 Symmetric Encryption Supporting Fragmentation

6.2.1 Unified Syntax

Morphology. We now extend the definition of symmetric encryption from Section 2.3.1 for the case of fragmented ciphertexts. As before we will have that

$\mathcal{M} \in \{0,1\}^*$ and $\mathcal{C} \in \{0,1\}^*$; in addition we will allow schemes to have multiple errors as in Chapter 5. Complications arise when considering encryption schemes in the presence of ciphertext fragmentation. For instance, a single ciphertext can be split up in multiple fragments (Fig. 6.1, requiring recombination of the decryptions of the various fragments) or a single fragment can contain multiple ciphertexts (Fig. 6.2, where one might like to decrypt to a list of messages).

Definition 6.1: Symmetric encryption scheme supporting fragmentation. A *symmetric encryption scheme supporting fragmentation* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with associated *message space* $\mathcal{M} \in \{0,1\}^*$, *ciphertext space* $\mathcal{C} = \{0,1\}^*$ and *error messages* $\mathcal{S}_\perp$ is defined by three algorithms:

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

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

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

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

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

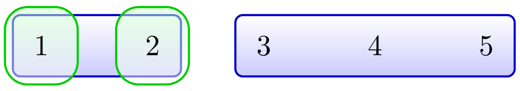

Figure 6.1: A single ciphertext $c_1 = (12)$ cut up in multiple fragments $f_1 = (1)$ and $f_2 = (2)$.

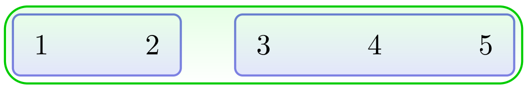

Figure 6.2: A single fragment $f_1 = (12345)$ spanning multiple ciphertexts $c_1 = (12)$ and $c_2 = (345)$.

Note that the decryption algorithm is assumed to be able to handle ciphertexts which decrypt to multiple plaintext messages, or to a mixture of plaintexts and error symbols, or possibly to nothing at all (perhaps because the input ciphertext is insufficient to enable decryption to yet output anything, giving a significant difference from the atomic setting where decryption always outputs something). We use the symbol $\P \notin \{0,1\} \cup \mathcal{S}_\perp$ to denote the end of plaintext messages, enabling an application making use of the decryption algorithm to parse the output uniquely into a sequence of elements of $\mathcal{M}$ and errors from $\mathcal{S}_\perp$.

While we enforce that from a decryption of a sequence of ciphertext fragments, the corresponding message boundaries are easy to distinguish, we make no such requirement for ciphertexts. Indeed, given a sequence of ciphertext fragments, it will not be a priori clear what the constituent ciphertexts are (and in fact, in Section 6.4, we want to model schemes which hide these boundaries as a security goal). Looking ahead, the absence of clear ciphertext boundaries (in a sequence of fragments) will cause challenging parsing problems for our CCA definitions: in order to ‘forbid’ decryption of the challenge ciphertext, a prerequisite is that this challenge ciphertext can be located accurately in the sequence of ciphertext fragments!

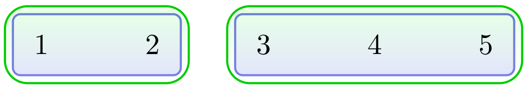

Figure 6.3: Fragments $f_1 = (12)$ and $f_2 = (345)$ coincide exactly with the two ciphertexts $c_1 = (12)$ and $c_2 = (345)$.

Correctness. If a single message is encrypted and the corresponding ciphertext is subsequently decrypted, we expect that the message is returned. When multiple messages are encrypted and the fragments correspond *exactly* to the ciphertext (Fig. 6.3), again we expect to retrieve the original messages.

However, we expect something stronger, namely that regardless of how we fragment the ciphertext(s), the original message(s) are returned. For instance in the situation depicted in Fig. 6.1 two ciphertexts $c_1 = (12)$ and $c_2 = (345)$ are produced by the encryption oracle, and the adversary subsequently submits fragments $f_1 = (1)$ and $f_2 = (2)$ to its decryption oracle. We require that after reception of the second fragment (or earlier), the ciphertext c_1 gets decrypted (formalised by the correct message being output). Similarly, in Fig. 6.2, after reception of the single fragment spanning two ciphertexts, we expect both messages to be returned (with correct message boundaries indicated).

Finally, we require correct decryption, even when an extra string is added to the original (string of) ciphertexts. This forces correct decryption once a complete valid ciphertext has been received, even if it is followed by an invalid ciphertext fragment. For instance, in the situation depicted in Fig. 6.5 two ciphertexts $c_1 = (12)$ and $c_2 = (345)$ are produced by the encryption oracle, the adversary subsequently submits fragments $f_1 = (1)$ and $f_2 = (234'5')$ to its decryption oracle, and we still want to see the first ciphertext decrypted properly.

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

Definition 6.2: Correctness Requirement. For all (K, σ_0, ϱ_0) that can be output by $\mathcal{K}$ and for all $\mathbf{m} \in \mathcal{M}^*$ and $\mathbf{f} \in (\{0,1\}^*)^*$, it holds (with probability 1) that if $(\mathbf{c}, \sigma) \leftarrow \mathcal{E}_K(\mathbf{m}, \sigma_0)$ and $||(\mathbf{c})$ prefixes $||(\mathbf{f})$, and if $(m', \varrho) \leftarrow \mathcal{D}_K(\mathbf{f}, \varrho_0)$ then m' is prefixed by $\P(\mathbf{m})$.

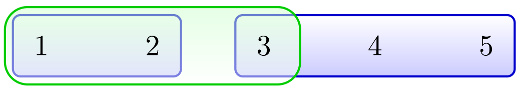

Figure 6.4: Correct decryption for overly long fragments: Given valid ciphertexts $c_1 = (12)$ and $c_2 = (345)$ and fragment $f_1 = (123)$, what should the decryption of f_1 be?

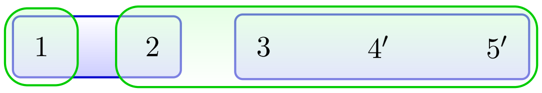

Figure 6.5: Two consecutive fragments $f_1 = (1)$ and $f_2 = (234'5')$. The second fragment completes the first ciphertext $c_1 = (12)$, so we expect that to be decrypted at this point, even though ciphertext $c_2 = (345)$ in the second fragment has been modified to produce a possibly invalid ciphertext.

Alternatives. Our choice for correctness (Definition 6.2) might seem natural, but it is not the only way to define it. Certainly, if a single, honestly generated ciphertext is cut up into multiple fragments, then decrypting all those fragments ought to result in the original message. This extends to a situation where (the concatenation of) multiple ciphertexts is split up into fragments in such a way that *every* ciphertext boundary coincides with a fragment boundary (this implies that every fragment is a substring of a single ciphertext). However, when allowing a single fragment to extend over multiple ciphertexts (see Fig. 6.4 for an example), it is not immediately clear what 'correct' entails. Let us briefly consider three possible interpretations.

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

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

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

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

6.2.2 Degrees of Statefulness

In Chapter 2 we considered stateful schemes as a special type of symmetric encryption schemes, and saw that such schemes are able to meet stronger notions of security. However when considering ciphertext fragmentation we see that schemes need to be stateful. Thus we will mainly consider stateful schemes, and as a special case we will consider schemes that employ only a 'minimal' form of state necessary to support fragmentation. For each notion of security we will first present a 'strong' variant following similar ideas to the security notions in Section 2.5. These notions will generally be met only by schemes which maintain a non-minimal state. Then we will present weakened variants of these notions which can be met by schemes within the special subclass having minimal state.

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

Stateless beyond buffering (sbb) schemes. This is a subclass of the above category, which is intuitively an extension of standard (atomic), stateless encryption schemes that makes handling fragmented ciphertexts possible. Namely, we specify three properties which intuitively capture the behaviour of a buffer, and require that the decryption state satisfy these properties. Our formulation allows us to identify states that essentially act as buffers, without imposing any restrictions on the state's internals or format. Again the encryption algorithm may or may not be stateful. This is in line with [16] where the statefulness of a scheme is determined solely by the statefulness of the decryption algorithm. More formally, we have:

Definition 6.3: Stateless beyond buffering (sbb). A symmetric encryption scheme supporting fragmentation is called *stateless beyond buffering* (or *sbb* for short) if it is correct (Definition 6.2) and satisfies the following additional conditions

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

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

6.3 Confidentiality

6.3.1 The Stateful Notion

When considering the security of a scheme supporting fragmentation, the first thing to note is that fragmentation matters only in the CCA setting: if there is no decryption oracle, then whether decryption is fragmented or atomic is immaterial to the security of the scheme. In the context of fragmentation, we will replace the usual notion of chosen-ciphertext attacks by chosen-fragment attacks (CFA). Our first notion, IND-sfCFA is tailored for stateful schemes and it is inspired by Bellare et al.'s [16] notion of IND-sfCCA (for atomic schemes) presented in Section 2.5. Recall that for IND-sfCCA, an adversary has unlimited access to the decryption oracle; there are no 'prohibited' queries. Instead, to avoid trivial attacks (by the adversary simply relaying its challenge ciphertext for decryption) a syncing mechanism is used. Initially the decryption oracle is in-sync and its output (to the adversary) will be suppressed. Only when the adversary causes the decryption oracle to be out-of-sync (by deviating from the ciphertext stream output by the encryption oracle) will the purported plaintexts (or error messages) be returned.

For atomic schemes, this is relatively straightforward to define, but for schemes supporting fragmentation, some ambiguity arises. Consider again the scenario sketched in Fig. 6.5. The first fragment is in-sync and any plaintext output corresponding to it will be suppressed. In the second fragment a deviation from the challenge ciphertext stream occurs. However, *part* of the fragment is still in-sync and certainly outputting the full decryption would—mindful of the correctness requirement—reveal (part of) the plaintext (12). We will need to formalise this by officially declaring part of the fragment in-sync, and part of it out-of-sync. The ambiguity arises with regards to the boundary we should use: is sync lost already at '3' (being the first symbol of a ciphertext that is not completed properly) or only at '4' (being the first symbol of the fragment that actually deviates)?

In our definition of IND-sfCFA (Definition 6.4) we opted for the strongest interpretation, namely where synchronisation is lost at the ciphertext boundary. Since this results in synchronization potentially being lost *earlier*, the decryption oracle consequently suppresses *less* of its output, making it the stronger option.

Definition 6.4: IND-sfCFA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a bit b, define experiment $\mathbf{Exp}_{\mathcal{SE}}^{\text{ind-sfcfa-b}}(\mathcal{A})$ as depicted in Fig. 6.6. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is given access to a left-or-right encryption oracle $\mathsf{LoR}(\cdot)$ and a stateful decryption oracle $\mathsf{sfDec}(\cdot)$. The stateful decryption oracle can be queried on any sequence of ciphertext fragments, but as long as the decryption queries are in sync the output will be artificially suppressed.

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

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

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

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

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

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

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

Figure 6.6: Experiment to define IND-sfCFA security.

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

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

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

Figure 6.7: Experiment to define IND-sbbCFA security.

6.3.2 A Notion for SBB Schemes

Similarly to the stateful notions from [16], Definition 6.4 protects against attacks which replay and reorder ciphertexts. In order for a scheme to protect against such attacks it needs to maintain a decryption state across ciphertexts. Hence IND-sfCFA is 'too strong' for SBB schemes, as the second requirement of Definition 6.3 explicitly rules out the ability to maintain states across ciphertexts. Accordingly for SBB schemes we propose an analogous but weaker notion of confidentiality which does not capture replay and reordering of ciphertexts. In this setting detecting prohibited queries, which would lead to trivial win conditions, becomes challenging. In fact we resort to specific properties of SBB schemes in this definition, specifically literal decryption and the property that no state is maintained across ciphertexts. Inside the decryption oracle we exploit the literal decryption property to decrypt ciphertext fragments incrementally, i.e. bit by bit. Then we can detect ciphertext boundaries by looking for a condition where the last output symbol is ¶ and the decryption state is empty. Since Definition 6.5 makes use of properties specific to SBB schemes, it only guarantees a meaningful notion of security for this subclass of schemes, and not schemes supporting fragmentation in general.

Definition 6.5: IND-sbbCFA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation that is stateless-beyond-buffering. For an adversary $\mathcal{A}$ and a bit b, define experiment $\mathbf{Exp}_{\mathcal{SE}}^{\text{ind-sbbcfa-b}}(\mathcal{A})$ as depicted in Fig. 6.7. The experiment start by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to a left-or-right encryption oracle $\mathsf{LoR}(\cdot)$ and a decryption oracle $\mathsf{Dec}(\cdot)$. The decryption oracle can be queried on any sequence of ciphertext fragments, except that the output corresponding to a ciphertext previously output by the left-or-right oracle will be artificially suppressed.

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

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

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

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

6.4 Boundary Hiding

It is conventional wisdom that an encryption scheme cannot hide entirely the message length from an adversary. In practice however, the message length can convey information about the nature of the message. For instance the IPsec attacks from Chapter 4 identify ICMP error messages from their length. As another example, traffic analysis has been used to derive approximate transcripts of encrypted Voice over IP (VoIP) conversations [96]. Traffic analysis is a real concern, and in practice heuristic countermeasures are commonly employed to mitigate such attacks. Practical protocols like TLS, SSH, and IPsec use variable-length padding, while IPsec additionally provides the ability to insert dummy messages/packets. A recent study by Dyer et al. [38] shows that none of the aforementioned countermeasures, together with others that have been proposed in the literature, are effective in preventing HTTP fingerprinting. However their attacks do not rely solely on ciphertext lengths.

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

We give definitions for both the passive and the active adversary cases. The passive case is the one that is commonly assumed in the traffic analysis literature [38, 96]. Here the adversary merely monitors encrypted traffic and tries to infer information from ciphertext lengths and other information such as network packet timings, but without giving away its presence by actively modifying network traffic. By hiding the ciphertext boundaries, the adversary can no longer determine individual ciphertext lengths, except of course, for the total volume being sent. As we will see, achieving security in the passive case is relatively straightforward. Much more challenging is achieving security in the active case. For example, it was already pointed out by Albrecht et al. [1] that SSH, while attempting to hide ciphertext boundaries, fails to do so against active, fragmented attacks (there is a simple bit-flipping attack which works irrespective of whether CBC or CTR mode encryption is used in the SSH construction).

6.4.1 Security Definitions

Definition 6.6: BH-CPA and BH-sfCFA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a bit b, define experiments $\mathbf{Exp}_{\mathcal{SE}}^{\text{bh-cpa-b}}(\mathcal{A})$ and $\mathbf{Exp}_{\mathcal{SE}}^{\text{bh-sfcfa-b}}(\mathcal{A})$ as shown in Figure 6.8. Both experiments start by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is given access to a special left-or-right encryption oracle $\mathsf{LoR}(\cdot)$: on input two vectors of messages, either the left or the right result is returned, but with the caveat that the concatenation of ciphertexts is returned *only* if it has the same length in both worlds (but note that we do not insist that the two vectors of messages contain the same number of components). In the latter experiment the adversary is additionally given a stateful decryption oracle $\mathsf{sfDec}(\cdot)$ identical to that used in the $\mathsf{IND\text{-}sfCFA}$ experiment. The adversary can query this oracle on any sequence of ciphertext fragments, but as long as the decryption queries are in sync the output is artificially suppressed.

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

  $(K, \sigma, \varrho) \leftarrow \mathcal{K}$
  $i \leftarrow 0, j \leftarrow 0, \mathsf{sync} \leftarrow 1$
  $C \leftarrow \varepsilon, F \leftarrow \varepsilon, M \leftarrow \varepsilon$
  $\mathtt{C} \leftarrow (), \mathtt{M} \leftarrow ()$
  $b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot)}$ $\boxed{b' \leftarrow \mathcal{A}^{\mathsf{LoR}(\cdot),\mathsf{sfDec}(\cdot)}}$
  return $b'$

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

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

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

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

Figure 6.8: Experiments to define BH-CPA and BH-sfCFA security. For BH-CPA the boxed code is excluded, whereas for BH-sfCFA the boxed code replaces the code adjacent to it.

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

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

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

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

Analogously to the case of confidentiality, we can define a natural SBB variant of this notion by replacing the stateful decryption oracle $\mathsf{sfDec}(\cdot)$ in Fig. 6.8 with the decryption oracle $\mathsf{Dec}(\cdot)$ from Fig. 6.7. As before the resulting experiment, displayed in Fig. 6.9, assumes properties that are specific to SBB schemes, and hence this security notion is only meaningful for SBB schemes.

Definition 6.7: BH-sbbCFA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation that is stateless-beyond-buffering. For an adversary $\mathcal{A}$ and a bit b, define the experiment $\mathbf{Exp}^{\text{ind-sbbcfa-b}}_{\mathcal{SE}}(\mathcal{A})$ as depicted in Fig. 6.9. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is given access to a special left-or-right encryption oracle $\mathsf{LoR}(\cdot)$: on input two vectors of messages, either the left or the right result is returned, but with the caveat that the concatenation of ciphertexts is returned *only* if it has the same length in both worlds (but note that we do not insist that the two vectors of messages contain the same number of components). The adversary is additionally given a decryption oracle $\mathsf{Dec}(\cdot)$. It can query the decryption oracle on any sequence of ciphertext fragments, except that the output corresponding to a ciphertext previously output by the left-or-right oracle will be artificially suppressed.

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

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

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

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

Figure 6.9: Experiment to define BH-sbbCFA security.

experiment returns b' as well. We define the advantage of an adversary $\mathcal{A}$ as:

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

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

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

Relating Boundary Hiding to Indistinguishability. We now establish a few relations between notions of boundary hiding and notions of indistinguishability which we will use in later Sections. Theorem 6.1 states that for length-regular[1] schemes boundary hiding implies left-or-right indistinguishability. Intuitively this follows because the special left-or-right oracle in the BH-ATK notions can be used to simulate the left-or-right oracle in the IND-ATK notions. The requirement on length-regularity ensures that a valid query to the IND-ATK oracle results in a valid query to the BH-ATK oracle. Other than that the proof is straightforward and we omit it. Moreover, it is not too hard to show that BH-ATK is strictly stronger than IND-ATK: take any IND-ATK secure scheme and append each ciphertext with a special marker string, e.g., 1^{128}.

Theorem 6.1: BH-ATK $\longrightarrow$ IND-ATK. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a length-regular symmetric encryption scheme supporting fragmentation. For any* ATK $\in$ {CPA, sbbCFA, sfCFA} *and any* IND-ATK *adversary $\mathcal{A}_{ind}$ there exists a* BH-ATK *adversary $\mathcal{A}_{bh}$ consuming similar resources to $\mathcal{A}_{ind}$ such that:*

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

Intuitively the concatenation of multiple random strings is indistinguishable from a

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

single random string of the same length. It then follows that schemes having ciphertexts indistinguishable from random strings should also hide ciphertext boundaries. This is stated more formally, for the passive setting[2], in the following theorem. Again it is not hard to show that this implication is strict: take any BH-CPA secure scheme and re-encode its ciphertexts by doubling every bit, i.e., $0 \to 00$ and $1 \to 11$.

Theorem 6.2: IND\$-CPA $\longrightarrow$ BH-CPA. *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme supporting fragmentation. For any* BH-CPA *adversary $\mathcal{A}_{bh}$ there exists an* IND\$-CPA *adversary $\mathcal{A}_{ind\$}$ consuming similar resources to $\mathcal{A}_{bh}$ such that:*

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

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

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

□

[2]It can be shown that this implication does not hold in the stateful setting, in fact the stateful InterMAC construction of Section 6.6.2 serves as a separating example.

6.4.2 Unsatisfiability of SBB Boundary Hiding

We now outline a general attack, applicable to any practically relevant scheme, showing that the BH-sbbCFA definition given in Figure 6.9 is unsatisfiable. The reader is recommended to first refer to the next section where security against Denial of Service attacks is introduced and defined.

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

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

6.5 Denial of Service

In this section we study fragmentation-related Denial-of-Service (DoS) attacks. This is, to the best of our knowledge, the first formal treatment of DoS prevention as a property of a symmetric encryption scheme. In Section 6.1 we mentioned an example of a fragmentation-related attack that constitutes DoS. In that attack, by carefully tampering with only a few bits in one of the transmitted ciphertexts, the adversary managed to 'confuse' the decryption algorithm so that it would produce no output until a huge amount of ciphertext is received. Informally this kind of attack is what our security notions will attempt to capture. More specifically, we will equip the adversary with an encryption oracle and a decryption oracle. Its goal will be to produce a sequence of ciphertext fragments whose concatenation is at least n bits long, where each of these fragments decrypts to the empty string. We will then quantify the DoS security of a scheme via the minimum value of n such that no 'efficient' adversary is successful in producing such a sequence of fragments.

The countermeasure adopted by SSH to mitigate against such attacks (see Section 3.2) is to limit the maximum ciphertext length to n bits; thereby ensuring that the decryption algorithm will produce an output after at most n bits of ciphertext. In the case of OpenSSH n is set to 2^{21}. We consider this to be a serious limitation since it affects the usability of the scheme. If two parties wish to exchange large files, it is understood that this may require waiting for large amounts of ciphertext before recovering it at the receiver side, and this should be allowed. What we wish to avoid is cases where the communicating parties are exchanging short messages, but the adversary is able to tamper with the ciphertexts in such a way that the receiver has to wait for a large amount of ciphertext before producing an output. Thus we aim to formulate DoS security in a way that allows lowering n without necessarily restricting the maximum message size in the message space. To accommodate this we exclude trivial win conditions of the type where a passive adversary forwards ciphertexts (or fragmentations thereof) of length n or higher from the encryption oracle to the decryption oracle. In essence we will insist that the sequence of fragments by which the adversary wins be generated by an active adversary. In the stateful setting this means that we will require the 'winning' sequence of fragments to occur after the adversary has become active. In the SBB setting, trivial win conditions will be trickier to catch. We now formulate the two definitions more precisely.

6.5.1 Security Definitions

Definition 6.8: n-DOS-sfCFA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a positive integer n, define the experiment $\mathbf{Exp}_{\mathcal{SE}}^{n\text{-dos-sfcfa}}(\mathcal{A})$ as depicted in Fig. 6.10. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$, and a stateful decryption oracle $\mathsf{sfDec}(\cdot)$. The adversary's goal is to submit to the stateful decryption oracle $\mathsf{sfDec}(\cdot)$ an out-of-sync sequence of fragments whose combined length is at least n bits, such that all fragments return no output upon decryption. In the event that the adversary succeeds, the experiment returns 1, and 0 otherwise. The output of the stateful decryption oracle is never suppressed.

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

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

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

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

We now define an analogous DoS security notion in the SBB setting. As before

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

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

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

Figure 6.10: Experiment to define n-DOS-sfCFA security.

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

Definition 6.9: n-DOS-sbbCFA. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme supporting fragmentation. For an adversary $\mathcal{A}$ and a positive integer n, define the experiment $\mathbf{Exp}^{n\text{-dos-sfcfa}}_{\mathcal{SE}}(\mathcal{A})$ as depicted in Fig. 6.11. The experiment starts by calling $\mathcal{K}$ to generate a key K and initialise the states. The adversary $\mathcal{A}$ is then given access to an encryption oracle $\mathsf{Enc}(\cdot)$, and a decryption oracle $\mathsf{Dec}(\cdot)$. The adversary's goal is to submit to the decryption oracle $\mathsf{Dec}(\cdot)$ a non-trivial sequence of fragments whose combined length is at least n bits, such that all fragments return no output upon decryption. In the event that the adversary succeeds, the experiment returns 1, and 0 otherwise. The output of the decryption oracle is never suppressed.

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

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

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

Once again we exploit literal decryption to decrypt ciphertext fragments incrementally. This allows the decryption oracle to detect ciphertext boundaries in order to filter out trivial win conditions. The variable F stores the concatenation of all ciphertext bits belonging to the current ciphertext. If the end of ciphertext is detected (by checking for the condition where $\varrho = \varepsilon$ **and** $m'[|m'|] = \P$), then F is reset. The

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

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

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

Figure 6.11: Experiment to define DOS-sbbCFA security.

variable q points to the end of the last received fragment within F that produced an output. Each time a fragment is received the decryption oracle checks whether the concatenation of fragments that did not produce an output, i.e. $F[q+1, |F|]$, is at least n bits long. If so it further verifies that after all possible replayed ciphertext prefixes are removed, it still is at least n bits long.

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

6.6 Constructions

6.6.1 Applying Instantaneously Decodable Postprocessing (IDP)

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

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

Definition 6.10: Instantaneous Decodability. An encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ with associated word space $\mathcal{W} \subseteq \{0,1\}^*$, is said to be instantaneously decodable if for all keys K that can be output by $\mathcal{K}_c$, it holds that:

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

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

Construction 6.1: The IDP Construction. Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with associated message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ be an instantaneously decodable encoding scheme with an associated word space that contains $\mathcal{C}$. Then the construction specified in Figure 6.12 yields an encryption scheme supporting fragmentation $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ with an associated message space $\mathcal{M}$. Furthermore if $\mathcal{SE}$ is stateless, then $\overline{\mathcal{SE}}$ is stateless

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

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

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

Figure 6.12: The constructed scheme $\overline{\mathcal{SE}}$ using instantaneously decodable post-pocessing.

beyond buffering.

Correctness of the constructed scheme $\overline{\mathcal{SE}}$ follows immediately from the correctness of $\mathcal{SE}$ and the instantaneous decodability of $\mathcal{ES}$. Furthermore if $\mathcal{SE}$'s decryption algorithm is stateless, the only state that $\overline{\mathcal{D}}$ maintains is the buffer α. Now the buffer is always initialised to ε, and it is easy to see that after decrypting any complete ciphertext, the buffer will always be empty. Finally, because all submitted fragments are appended to the buffer from which ciphertexts are then extracted and submitted to $\mathcal{D}$, decryption is independent of the fragmentation pattern. Hence the scheme also satisfies literal decryption, and consequently is stateless beyond buffering.

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

Theorem 6.3: IDP is IND-sfCFA secure. *Let $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the scheme from Construction 6.1, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and an instantaneously decodable encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$. Then for any* IND-sfCFA *adversary $\mathcal{A}_{sfcfa}$ against $\overline{\mathcal{SE}}$, there exists an* IND-sfCCA *adversary $\mathcal{A}_{sfcca}$*

against $\mathcal{SE}$ such that:

$$\mathbf{Adv}^{\text{ind-sfcfa}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{sfcfa}) \leq \mathbf{Adv}^{\text{ind-sfcca}}_{\mathcal{SE}}(\mathcal{A}_{sfcca})\,, \tag{6.1}$$

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

Proof. For any adversary $\mathcal{A}_{sfcfa}$ we construct adversary $\mathcal{A}_{sfcca}$ as follows. Adversary $\mathcal{A}_{sfcca}$ runs $\mathcal{K}_c$ to generate an encoding key and then runs $\mathcal{A}_{sfcfa}$. It then uses the encoding key together with its left-or-right oracle to simulate $\mathcal{A}_{sfcfa}$'s left-or-right oracle as per Construction 6.1, keeping record of the ciphertexts it returns. $\mathcal{A}_{sfcfa}$'s decryption queries are handled by maintaining a buffer to which the queried fragments are appended. Then $\mathcal{A}_{sfcca}$ repeatedly applies the decoding algorithm to the buffer until no codeword can be extracted, and submits the codewords, in the same order, to its own stateful decryption oracle. When the queries become out of sync, the returned messages are appended with ¶, concatenated together, and the resulting string is returned to $\mathcal{A}_{sfcfa}$. $\mathcal{A}_{sfcca}$ uses its records to keep track of when $\mathcal{A}_{sfcfa}$'s queries become out of sync. This is necessary since the first out of sync query might correspond to an encryption of ε, and $\mathcal{A}_{sfcfa}$ would not be able to distinguish this case from a message being suppressed because it is in sync. Finally $\mathcal{A}_{sfcca}$ outputs whatever $\mathcal{A}_{sfcfa}$ outputs.

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

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

and equation (6.1) follows. □

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

Theorem 6.4: IDP is IND-sbbCFA secure. *Let $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the scheme from Construction 6.1, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$*

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

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

Figure 6.13: Experiment to define randomness preserving encodings.

with stateless decryption, and an instantaneously decodable encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$. Then for any IND-sbbCFA *adversary $\mathcal{A}_{sbbcfa}$ against $\overline{\mathcal{SE}}$, there exists an* IND-CCA *adversary $\mathcal{A}_{cca}$ against $\mathcal{SE}$ such that:*

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

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

Construction 6.1 shows that IND-sfCFA and IND-sbbCFA security are not hard to attain. However when we instantiate the encoding scheme with a prefix free encoding, ciphertext boundaries will inevitably be revealed. While this is what allows the constructed scheme to support ciphertext fragmentation, it obviously conflicts with the goal of boundary hiding. We partly solve this conflict by employing a keyed encoding scheme, which reveals ciphertext boundaries solely to the holder of the encoding key. We now formulate a security property for encoding schemes that will allow Construction 6.1 to achieve IND\$-CPA security, and by Theorem 6.2 hide ciphertext boundaries from passive adversaries.

Definition 6.11: Randomness Preserving Encodings. Let $\mathcal{L}$ be a non-empty set of positive integers, and let $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ be an encoding scheme with associated word space $\mathcal{W} = \bigcup_{\ell \in \mathcal{L}} \{0,1\}^\ell$. For an adversary $\mathcal{A}$ and a bit b define the experiment $\mathbf{Exp}^{\text{rpe-b}}_{\mathcal{ES}}(\mathcal{A})$ as shown in Figure 6.13. The experiment starts by calling $\mathcal{K}_c$ to generate an encoding key K_c. The adversary $\mathcal{A}$ is then given access to an encode-or-random oracle $\mathsf{EoR}(\cdot)$, that it can query on any length value $\ell \in \mathcal{L}$. Depending on the value of b, the oracle will either return an encoding of a random string of length ℓ, or a random string of the same length as that encoding. The adversary's goal is to output a bit b', as its guess of the challenge bit b, and the experiment returns b' as well. We define the adversary's rpe-advantage as:

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

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

Theorem 6.5: IDP is $\mathsf{IND\$}$-$\mathsf{CPA}$ secure. *Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with associated message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ be an encoding scheme with an associated word space that contains $\mathcal{C}$. Define the encryption scheme supporting fragmentation $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ according to Construction 6.1. For any* IND\$-CPA *adversary $\mathcal{A}_{ind\$}$ against $\overline{\mathcal{SE}}$, there exist adversaries $\mathcal{A}'_{ind\$}$ and $\mathcal{A}_{rpe}$ such that:*

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}'_{ind\$}) + \mathbf{Adv}^{\text{rpe}}_{\mathcal{ES}}(\mathcal{A}_{rpe})\,, \tag{6.2}$$

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

Proof. To prove Theorem 6.5 we introduce we introduce a hybrid experiment $\mathbf{ExpH}$, similar in spirit to the two IND\$-CPA experiments corresponding to each bit value. The hybrid experiment proceeds exactly as $\mathbf{Exp}^{\text{ind\$-cpa-1}}_{\overline{\mathcal{SE}}}$, except that the encryption oracle returns encodings of random strings instead. More specifically, after computing an encryption under $\mathcal{SE}$ of the queried message, it picks uniformly at random a string of the same length as the ciphertext, and returns an encoding under $\mathcal{ES}$ of this string instead. We then have that:

$$\begin{aligned}\mathbf{Adv}^{\text{ind\$-cpa}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) = {}& \Big(\Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-1}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right]\Big) \\ & + \Big(\Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-0}}_{\overline{\mathcal{SE}}}(\mathcal{A}_{ind\$}) = 1\,\right]\Big) . \end{aligned} \tag{6.3}$$

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

$$\Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-1}}_{\mathcal{SE}}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right] \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}'_{ind\$}) \tag{6.4}$$

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

$$\Pr\left[\,\mathbf{ExpH}(\mathcal{A}_{ind\$}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind\$-cpa-0}}_{\mathcal{SE}}(\mathcal{A}_{ind\$}) = 1\,\right] \leq \mathbf{Adv}^{\text{rpe}}_{\mathcal{ES}}(\mathcal{A}_{rpe})\,. \tag{6.5}$$

Combining equations (6.3),(6.4), and (6.5) yields equation (6.2), as desired. □

We now complete the IDP construction by showing a simple instantiation of an encoding scheme that is both instantaneously decodable and randomness preserving. The encoding scheme is constructed from a pseudorandom function family mapping n bit strings to l bit strings, and is presented in Figure 6.14.

Theorem 6.6: IDP Instantiation. *Let $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^l$ be a function family indexed by the set $\mathcal{K}$. Then Figure 6.14 defines an instantaneously decodable encoding scheme $\mathcal{ES} = (\mathcal{K}_c, \mathcal{EC}, \mathcal{DC})$ with word space $\mathcal{W} = \bigcup_{\ell \leq l}\{0,1\}^{\ell}$. Moreover, for any RPE adversary $\mathcal{A}_{rpe}$ against $\mathcal{ES}$ making at most q queries, there exists a PRF adversary $\mathcal{A}_{prf}$ such that:*

$$\mathbf{Adv}^{\text{rpe}}_{\mathcal{ES}}(\mathcal{A}_{rpe}) \leq \mathbf{Adv}^{\text{prf}}_{F}(\mathcal{A}_{prf}) + \left(\frac{q^2}{2^{n+1}}\right)\,, \tag{6.6}$$

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

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

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

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

Figure 6.14: The encoding scheme of Theorem 6.6 that is both instantaneously decodable and randomness preserving.

Proof. We first outline why the encoding scheme is instantaneously decodable. Note that the decoding algorithm first recovers the length field and then uses this value to determine where the codeword ends. Thus the first requirement of Definition 6.10 is satisfied. As for the second requirement, note that the only case where the input string s is not prefixed by a valid code word is either when its length is less than or equal to $n + l$, or the recovered length value is greater than the length of the remaining string. In both cases the decoding algorithm returns (ε, s), as required.

We now prove that the encoding scheme is randomness preserving. To do this, we show that from any RPE adversary $\mathcal{A}_{rpe}$, we can build a PRF adversary $\mathcal{A}_{prf}$ against F. Adversary $\mathcal{A}_{prf}$ runs $\mathcal{A}_{rpe}$, and simulates its oracle by sampling random strings of the queried length and encoding them according to the construction of Figure 6.14 and computing PRF values using its own oracle. $\mathcal{A}_{prf}$ keeps a record of all the n bit strings that it samples, and if at any point a collision occurs it outputs 0 and halts. Otherwise $\mathcal{A}_{prf}$ outputs whatever $\mathcal{A}_{rpe}$ outputs. Note that when $\mathcal{A}_{prf}$ is instantiated with F, it responds to $\mathcal{A}_{rpe}$'s queries with real encodings of random strings. On the other hand if its oracle is a random function it returns uniformly random strings (since it never queries its oracle on the same value more than once). Let Z^b represent the event $\mathbf{Exp}^{\text{rpe-b}}_{\mathcal{ES}}(\mathcal{A}_{rpe}) = 1$, and let E represent the event that a collision occurs when sampling n bit strings. Then we have that:

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

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

$$\begin{aligned}\mathbf{Adv}^{\mathrm{rpe}}_{\mathcal{ES}}(\mathcal{A}_{rpe}) &\leq \Pr\left[\, K \leftarrow_{\$} \mathcal{K} : \mathcal{A}_{prf}^{F_K(\cdot)} = 1 \,\right] \\ &\quad - \Pr\left[\, f \leftarrow_{\$} \mathsf{Func}(n,l) : \mathcal{A}_{prf}^{f(\cdot)} = 1 \,\right] + \left(\frac{q^2}{2^{n+1}}\right). \qquad (6.7)\end{aligned}$$

Equation (6.6) then follows from equation (6.7). □

The IDP construction is attractive in terms of efficiency, modularity, and versatility. If we look at prior constructions, we see that achieving confidentiality and hiding boundaries while supporting fragmentation was already a source of conflict. Consider SSH for instance. Its effort to encrypt the length field can be interpreted as an attempt to hide boundaries. When instantiated with CBC encryption, it is easy to see that SSH achieves BH-CPA security, but as evidenced by the attack from [1] (see Section 3.5) it is insecure in the IND-sfCFA sense. Alternatively if we look at TLS, the result of [78] implies that it is IND-sfCCA secure. Moreover the length field contained in the header works as a prefix free encoding, and therefore by Theorem 6.3, TLS is IND-sfCFA secure. However since the header is in cleartext the scheme obviously does not achieve BH-CPA security.

6.6.2 The InterMAC Construction

We now move to a more ambitious goal, to simultaneously achieve all three of our security notions. In comparison to the IDP construction and SSH, we now additionally consider boundary hiding against active adversaries and DoS security. None of the schemes considered thus far achieve boundary hiding in the active setting. To see the difficulty with this consider once more the case of SSH. Given a concatenation of ciphertexs, the adversary can now flip the first bit and submit it bit by bit to its decryption oracle until an error is returned, which marks the first ciphertext boundary. In addition to achieving boundary hiding security in the active setting, the

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

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

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

Figure 6.15: The stateful InterMAC construction $\mathcal{IM}$.

scheme that we present in this section also achieves N-$\mathsf{DOS\text{-}sfCFA}$ security *without* limiting the maximum message size to N bits.

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

Construction 6.2: InterMAC. Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme such that its message space contains $\{0,1\}^{\ell_m+1}$, for some desired $\ell_m \in \mathbb{N}$. Furthermore let $\mathcal{E}$ be length-regular, such that it maps all messages of length ℓ_m to ciphertexts of length ℓ_c. Let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a message authentication code with associated tag length ℓ_{tag} and message space $\{0,1\}^*$. Then the stateful InterMAC construction, specified in Figure 6.15, yields an encryption scheme supporting fragmentation $\mathcal{IM} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ with message space $\{\{0,1\}^{\ell_m}\}^+$. The ciphertext segment size N associated to the stateful InterMAC construction is given by $N = \ell_c + \ell_{tag}$.

At first sight Figure 6.15 may seem daunting. Accordingly we now give an informal description. Each message is split into chunks of ℓ_m bits, a bit is then prepended to each chunk and encrypted separately. For all chunks of plaintext except the last, the prepended bit is set to zero. For each of these ciphertexts c', a MAC tag is computed over the concatenation of the encoded message counter $\langle i \rangle$, the encoded segment index $\langle j \rangle$, and the ciphertext. These ciphertext-tag pairs are then concatenated to yield the final ciphertext. Decryption starts by appending the input ciphertext fragment f to the buffer string α, and resetting the output plaintext string w. The while loop then extracts ciphertext segments from the buffer one at a time. Each segment is parsed into a ciphertext and a MAC tag, and the tag is then verified. The returned output string w is then constructed as follows. For valid ciphertexts, i.e. ciphertexts where all segments contain a valid tag, the plaintext is only returned when the last ciphertext segment has been received. Alternatively, once an invalid segment is encountered, the $\perp$ symbol is returned for that segment and every segment (irrespective of its validity) that is received thereafter.

Theorem 6.7: InterMAC is N-DOS-sfCFA secure. *Let $\mathcal{IM} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the InterMAC scheme from Construction 6.2, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Let its segment size be N. Then for any N-DOS-sfCFA adversary $\mathcal{A}_{dos}$ against $\mathcal{IM}$, there exists a UF-CMA adversary $\mathcal{A}_{uf}$ against $\mathcal{MA}$ such that:*

$$\mathbf{Adv}^{N\text{-dos-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{dos}) \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf})\,, \tag{6.8}$$

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

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

Adversary $\mathcal{A}_{uf}$ runs $\mathcal{K}_e$ to get an encryption key and initialise the states. It then runs $\mathcal{A}_{dos}$ and uses the encryption key together with its tagging oracle to simulate $\mathcal{A}_{dos}$'s encryption oracle (as per Construction 6.2). In addition it maintains an ordered list of all the ciphertexts it returns, together with their corresponding messages. $\mathcal{A}_{dos}$'s decryption queries are then handled as follows. $\mathcal{A}_{uf}$ maintains the string F^* (as defined above), and uses it together with the other list to keep track of when $\mathcal{A}_{dos}$ becomes active. Moreover it maintains the decryption counters i and j (as per Construction 6.2). While $\mathcal{A}_{dos}$'s queries are in sync, it uses its list to simulate the decryption oracle. When it happens that $|F^*| \geq (u+1)N$, it parses $F^*[uN+1,(u+1)N]$ into a ciphertext c and a MAC tag τ, submits the pair $(\langle i \rangle \parallel \langle j \rangle \parallel c, \tau)$ to its verification oracle, and halts.

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

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

and equation (6.8) thus follows. □

Note that we only have BH-sfCFA security left to prove, since IND-sfCFA security will then be implied by Theorem 6.1.

Theorem 6.8: InterMAC is BH-sfCFA secure. *Let $\mathcal{IM} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the InterMAC scheme from Construction 6.2, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Then*

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

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

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

Figure 6.16: The auxiliary experiment used to prove Theorem 6.8.

for any BH-sfCFA *adversary $\mathcal{A}_{sfcfa}$ against $\mathcal{IM}$, there exists adversaries $\mathcal{A}_{cpa}$, $\mathcal{A}_{prf}$, and $\mathcal{A}_{uf}$ such that:*

$$\frac{1}{2} \cdot \mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) , \quad (6.9)$$

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

Proof. We will prove Theorem 6.8 in two parts. For the first part of the proof we will make use of the auxiliary experiment $\mathbf{ExpA}^{\mathrm{b}}_{\mathcal{IM}}$ of Figure 6.16. This is essentially the $\mathbf{Exp}^{\text{bh-sfcfa-b}}_{\mathcal{IM}}$ experiment with a modified stateful decryption oracle. Now the stateful decryption oracle does not return any output until the queries become out of sync, at which point it returns $\perp$ at every N-bit boundary of ciphertext that it receives. At any point in time, let F^* be the concatenation of all ciphertext fragments queried by $\mathcal{A}_{sfcfa}$, and let u be the largest non-negative integer such that the substring $F^*[1, uN]$ is in sync. Let E represent the event that in the BH-sfCFA experiment $|F^*| \geq (u+1)N$ and $\mathsf{sfDec}(\cdot)$ did not return $\perp$ after receiving the first $((u+1)N)$ bits. Let W denote the event $\mathbf{Exp}^{\text{bh-sfcfa-b}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) = b$ and let W^A denote the event $\mathbf{ExpA}^{\mathrm{d}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) = d$, where bits b and d are picked uniformly at random.

We thus have that:

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

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

$$\begin{aligned}\Pr[\,W\,] - \Pr\left[\,W^A\,\right] &\leq \Pr[\,W \wedge E\,] + \Pr\left[\,W \mid \overline{E}\,\right] - \Pr\left[\,W^A\,\right] \\ &\leq \Pr[\,W \wedge E\,] \\ &\leq \Pr[\,E\,] . \qquad (6.10)\end{aligned}$$

Using a reduction similar to that in the proof of Theorem 6.7, it follows that there exists a UF-CMA adversary $\mathcal{A}_{uf}$ such that:

$$\Pr[\,E\,] \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) . \qquad (6.11)$$

Combining equations (6.10) and (6.11), and then multiplying by two and subtracting one on each side of the inequality, yields:

$$\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) \leq \left(2 \cdot \Pr\left[\,W^A\,\right] - 1\right) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) . \qquad (6.12)$$

Now from any adversary $\mathcal{A}_{sfcfa}$, we can construct a BH-CPA adversary $\mathcal{A}''_{cpa}$ against $\mathcal{IM}$ as follows. $\mathcal{A}''_{cpa}$ runs $\mathcal{A}_{sfcfa}$, and forwards its encryption queries to its own encryption oracle while keeping record of all ciphertexts that it returns. Decryption queries are handled by running the $\mathsf{sfDec}(\cdot)$ algorithm of Figure 6.16. Finally $\mathcal{A}''_{cpa}$ outputs whatever $\mathcal{A}_{sfcfa}$ outputs. Note that $\mathcal{A}''_{cpa}$ provides $\mathcal{A}_{sfcfa}$ with a perfect simulation of the auxiliary experiment. It then follows that:

$$\Pr\left[\,W^A\,\right] = \Pr\left[\,d \leftarrow_{\$} \{0,1\} \;:\; \mathbf{Exp}^{\text{bh-cpa-d}}_{\mathcal{IM}}(\mathcal{A}''_{cpa}) = d\,\right] . \qquad (6.13)$$

Combining equations (6.12) and (6.13), we obtain:

$$\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) \leq \mathbf{Adv}^{\text{bh-cpa}}_{\mathcal{IM}}(\mathcal{A}''_{cpa}) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) , \qquad (6.14)$$

and then applying Theorem 6.2 yields:

$$\mathbf{Adv}^{\text{bh-sfcfa}}_{\mathcal{IM}}(\mathcal{A}_{sfcfa}) \leq 2 \cdot \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{IM}}(\mathcal{A}'_{cpa}) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) . \qquad (6.15)$$

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

$$\begin{aligned}\mathbf{Adv}_{\mathcal{IM}}^{\text{ind\$-cpa}}(\mathcal{A}'_{cpa}) &= \left(\Pr\left[\,\mathbf{Exp}_{\mathcal{IM}}^{\text{ind\$-cpa-1}}(\mathcal{A}'_{cpa}) = 1\,\right] - \Pr\left[\,\mathbf{ExpH}(\mathcal{A}'_{cpa}) = 1\,\right]\right) \\ &+ \left(\Pr\left[\,\mathbf{ExpH}(\mathcal{A}'_{cpa}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}_{\mathcal{IM}}^{\text{ind\$-cpa-0}}(\mathcal{A}'_{cpa}) = 1\,\right]\right) . \quad (6.16)\end{aligned}$$

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

$$\Pr\left[\,\mathbf{Exp}_{\mathcal{IM}}^{\text{ind\$-cpa-1}}(\mathcal{A}'_{cpa}) = 1\,\right] - \Pr\left[\,\mathbf{ExpH}(\mathcal{A}'_{cpa}) = 1\,\right] \leq \mathbf{Adv}_{\mathcal{T}}^{\text{prf}}(\mathcal{A}_{prf}) . \quad (6.17)$$

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

$$\Pr\left[\,\mathbf{ExpH}(\mathcal{A}'_{cpa}) = 1\,\right] - \Pr\left[\,\mathbf{Exp}_{\mathcal{IM}}^{\text{ind\$-cpa-0}}(\mathcal{A}'_{cpa}) = 1\,\right] \leq \mathbf{Adv}_{\mathcal{SE}}^{\text{ind\$-cpa}}(\mathcal{A}_{cpa}) . \quad (6.18)$$

Combining equations (6.15),(6.16),(6.17), and (6.18) yields (6.9), as desired. □

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

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

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

Figure 6.17: The stateless beyond buffering InterMAC construction $\mathcal{IM}^*$.

6.6.3 A SBB Variant of InterMAC

Construction 6.3: SBB InterMAC. Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with stateless decryption, having a message space containing $\{0,1\}^{\ell_m+1}$ for some desired $\ell_m \in \mathbb{N}$, and error space $\mathcal{S}_\perp$. Furthermore let $\mathcal{E}$ be length-regular, such that it maps all messages of length ℓ_m to ciphertexts of length ℓ_c. Let $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$ be a message authentication code with associated tag length ℓ_{tag} and message space $\{0,1\}^*$. Let $\oslash$ be such that $\oslash \notin \mathcal{S}_\perp$. Then the SBB InterMAC construction, specified in Figure 6.17, yields an SBB encryption scheme supporting fragmentation $\mathcal{IM}^* = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ with message space $\{\{0,1\}^{\ell_m}\}^+$. The ciphertext segment size N associated to the stateful InterMAC construction is given by $N = \ell_c + \ell_{tag}$.

The above construction works analogously to its stateful counterpart, with a few exceptions. Counters are no longer maintained, and are therefore not included in the MAC tag computation and verification. Instead, the tag of the previous segment is prepended to the ciphertext when computing the MAC tag. For the purpose of computing the tag in the first segment of each ciphertext, the previous tag value is set to $0^{\ell_{tag}}$. In decryption, the fail flag has been dropped, and we now set m to the special symbol $\oslash$ instead. Thus as before, once an invalid MAC tag is detected, the decryption algorithm always returns $\perp$ from that point onwards. Note that this does not violate the SBB definition, see Section 6.2.2. Finally, the construction assumes an invertible encoding mapping the triple (α, m, τ_0) to a single string, such that $(\varepsilon, \varepsilon, 0^{\ell_{tag}})$ is mapped to the empty string. This technicality is required for the scheme to satisfy the SBB definition. Note that the decryption state does not contain more information than a buffer storing all bits pertaining to the ciphertext being decrypted. In fact it would have been functionally equivalent to let the decryption state be such a buffer, flushed only when the end of a ciphertext is found, and compute (α, m, τ_0) from this buffer each time the decryption algorithm is invoked. However we chose this implementation since it is less wasteful in computational resources, and yet satisfies the SBB definition.

Theorem 6.9: SBB InterMAC is N-DOS-sbbCFA secure. *Let $\mathcal{IM}^* = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the SBB InterMAC scheme from Construction 6.3, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ with stateless decryption and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Let its segment size be N. Then for any N-DOS-sbbCFA adversary $\mathcal{A}_{dos}$ against $\mathcal{IM}^*$ whose encryption queries total at most μ_e bits, there exist adversaries $\mathcal{A}_{uf}$ and $\mathcal{A}_{prf}$ such that:*

$$\mathbf{Adv}^{N\text{-dos-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{dos}) \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}}}\right), \quad (6.19)$$

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

Proof. Consider the experiment $\mathbf{Exp}^{N\text{-dos-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{dos})$. Let F and C be as in Figure 6.11, and let u be the largest non-negative integer such that there exists a $c \in \mathsf{C}$ satisfying $F[1, uN] \diamond c = F[1, uN]$. Let E represent the event that $|F| \geq (u+1)N$ and $\mathsf{Dec}(\cdot)$ did not return any output after receiving the first $((u+1)N)$ bits. For the case of InterMAC, if the adversary wins the experiment, then E must have occurred. Furthermore, let Q represent the event that for any two ciphertexts c and c' returned by the encryption oracle before E has occurred, there exist positive integers

x and y, where $x \leq y$, such that $c[xN - \ell_{tag} + 1, xN] = c'[yN - \ell_{tag} + 1, yN]$ but $c[1, xN] \neq c'[1, yN]$, or $c[xN - \ell_{tag} + 1, xN] = 0^{\ell_{tag}}$. Thus Q represents the event that either two tags collide or a tag value of all zeros occurs. We then have that:

$$\Pr\left[\,\mathbf{Exp}^{N\text{-dos-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{dos})) = 1\,\right] = \Pr[\,E\,] = \Pr[\,E \wedge Q\,] + \Pr\left[\,E \wedge \overline{Q}\,\right],$$

$$\leq \Pr[\,Q\,] + \Pr\left[\,E \mid \overline{Q}\,\right]. \qquad (6.20)$$

We now bound the probability of event Q occurring. Towards this goal we construct from $\mathcal{A}_{dos}$ a PRF adversary $\mathcal{A}_{prf}$ against $\mathcal{T}$. It starts by running $\mathcal{K}_e$ to get an encryption key and initialise the states. It then runs $\mathcal{A}_{dos}$ and uses the encryption key together with its oracle to simulate $\mathcal{A}_{dos}$'s encryption oracle (as per Construction 6.3). In addition it maintains a list of all the ciphertexts it returns, together with their corresponding messages. Note that by assumption event E has not occurred yet, thus $\mathcal{A}_{prf}$ is able to simulate $\mathcal{A}_{dos}$'s decryption oracle by using this list. Therefore, when $\mathcal{A}_{prf}$'s oracle is instantiated with $\mathcal{T}$ it provides $\mathcal{A}_{dos}$ with a perfect simulation of its environment. $\mathcal{A}_{prf}$ runs until $\mathcal{A}_{dos}$ halts or E occurs, at which point it checks whether Q has occurred. If so it outputs 1 otherwise it outputs 0. $\mathcal{A}_{prf}$ can check for Q as it proceeds by maintaining a list of the strings which it queried to its oracle, indexed by the returned tag values, and check for collisions or tag values of $0^{\ell_{tag}}$ while it populates the list. Consider now the case where $\mathcal{A}_{prf}$'s oracle is a random function. The probability of a collision in the tags can be bounded using a standard birthday bound, while the probability of a tag value of $0^{\ell_{tag}}$ is given by the number of queries divided by $2^{\ell_{tag}}$. Applying the union bound on these two probabilities, we have that:

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

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

$$\Pr[\,Q\,] \leq \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}}}\right). \qquad (6.21)$$

We now bound the second term of inequality (6.20), by constructing a $\mathsf{UF\text{-}CMA}$ adversary $\mathcal{A}_{uf}$ against $\mathcal{MA}$ from $\mathcal{A}_{dos}$. Adversary $\mathcal{A}_{uf}$ proceeds similarly to $\mathcal{A}_{prf}$. It runs $\mathcal{K}_e$ and uses this key together with its tagging oracle to simulate $\mathcal{A}_{dos}$'s encryption oracle. It also maintains a list of all the ciphertexts it returns together with their corresponding messages, and uses this to simulate $\mathcal{A}_{dos}$'s decryption oracle. It then keeps on simulating $\mathcal{A}_{dos}$'s environment until it halts or

$|F| \geq (u+1)N$. If $|F| \geq (u+1)N$ happens with $u > 0$, it submits the pair $(F[uN - \ell_{tag} + 1, uN + \ell_c], F[uN + \ell_c + 1, (u+1)N])$ to its verification oracle, and halts. Alternatively, if $|F| \geq (u+1)N$ occurs with $u = 0$, it submits the pair $(0^{\ell_{tag}} \parallel F[1, \ell_c], F[\ell_c + 1, N])$ instead. Note that until it occurs that $|F| \geq (u+1)N$, $\mathcal{A}_{uf}$'s simulation of $\mathcal{A}_{dos}$'s environment is perfect. Furthermore, if Q did not occur, it follows that the first component of the submitted pair was not previously queried to the tagging oracle. Thus assuming Q did not occur, whenever E occurs, $\mathcal{A}_{uf}$'s submitted pair constitutes a valid forgery. Therefore:

$$\Pr\left[\, E \mid \overline{Q} \,\right] \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf})\,. \tag{6.22}$$

Combining equations (6.20),(6.21) and (6.22), yields (6.19), as desired. □

A slightly different analysis could be used to achieve a possibly better bound for Theorem 6.9. In particular when bounding event Q, we could have considered the probability that both the tag and ciphertext values collide. This would lower the birthday bound term, at the expense of introducing an extra term of the form $\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathcal{SE}}(\mathcal{A})$. If the birthday bound is the dominant term, such an approach would yield a tighter bound. However we opted for a simpler proof of security.

Theorem 6.10: SBB InterMAC is IND-sbbCFA secure. *Let $\mathcal{IM}^* = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the SBB InterMAC scheme from Construction 6.3, composed from a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ with stateless decryption and message authentication code $\mathcal{MA} = (\mathcal{K}_m, \mathcal{T}, \mathcal{V})$. Let its segment size be N. Then for any* IND-sbbCFA *adversary $\mathcal{A}_{sbbcfa}$ against $\mathcal{IM}^*$ whose encryption queries total at most μ_e bits, there exist adversaries $\mathcal{A}_{cpa}$, $\mathcal{A}_{prf}$ and $\mathcal{A}_{uf}$, such that:*

$$\mathbf{Adv}^{\text{ind-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) \leq \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) + 2 \cdot \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}-1}}, \tag{6.23}$$

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

Proof. The proof of Theorem 6.10 follows the same lines as its stateful analogue. For the first part of the proof we will make use of the auxiliary experiment $\mathbf{ExpA}^{\text{b}}_{\mathcal{IM}}$ of

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

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

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

Figure 6.18: The auxiliary experiment used to prove Theorem 6.10.

Figure 6.18. This is essentially the $\mathbf{Exp}^{\text{ind-sbbcfa-b}}_{\mathcal{IM}^*}$ experiment, with the difference that once the decryption oracle detects a ciphertext which is not a replay of another ciphertext output by the encryption oracle, it then returns $\perp$ at every N-bit boundary of ciphertext that it receives. Now let F and C be as in Figure 6.7, and let u be the largest non-negative integer such that there exists a $c \in \mathsf{C}$ satisfying $F[1, uN] \diamond c = F[1, uN]$. Let E represent the event that in the IND-sbbCFA experiment $|F| \geq (u+1)N$ and $\mathsf{Dec}(\cdot)$ did not return $\perp$ after receiving the first $((u+1)N)$ bits. Let W denote the event $\mathbf{Exp}^{\text{ind-sbbcfa-b}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) = b$ and let W^A denote the event $\mathbf{ExpA}^{\mathrm{d}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) = d$, where bits b and d are picked uniformly at random. We then have that:

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

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

$$\begin{aligned} \Pr[W] - \Pr[W^A] &\leq \Pr[W \wedge E] + \Pr[W \mid \overline{E}] - \Pr[W^A] \\ &\leq \Pr[W \wedge E] \leq \Pr[E] . \end{aligned} \tag{6.24}$$

Using a reduction similar to that in the proof of Theorem 6.9, it follows that there exist adversaries $\mathcal{A}_{uf}$ and $\mathcal{A}_{prf}$ such that:

$$\Pr[\,E\,] \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) + \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}}}\right) . \tag{6.25}$$

Combining equations (6.24) and (6.25), and manipulating terms, yields:

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

Multiplying both sides by two and subtracting one:

$$\begin{aligned}\mathbf{Adv}^{\text{ind-sbbcfa}}_{\mathcal{IM}^*}(\mathcal{A}_{sbbcfa}) \leq \big(2 \cdot \Pr\left[\,W^A\,\right] - 1\big) + 2 \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(\mathcal{A}_{uf}) \, . \\ + 2 \cdot \mathbf{Adv}^{\text{prf}}_{\mathcal{T}}(\mathcal{A}_{prf}) + \left(\frac{\mu_e^2}{\ell_m^2 \cdot 2^{\ell_{tag}-1}}\right)\end{aligned} \tag{6.26}$$

Now from any adversary $\mathcal{A}_{sbbcfa}$, we can construct an IND-CPA adversary $\mathcal{A}_{cpa}$ against $\mathcal{SE}$ as follows. Adversary $\mathcal{A}_{cpa}$ runs $\mathcal{K}_m$ to obtain a key for the MAC and then runs $\mathcal{A}_{sbbcfa}$. It simulates its encryption oracle in accordance with the InterMAC scheme, except that it uses its own oracle to compute encryptions under $\mathcal{SE}$. In addition it maintains a list of all ciphertexts that it returns. Decryption queries are handled by simulating the $\mathsf{Dec}(\cdot)$ oracle of Figure 6.18. Finally $\mathcal{A}_{cpa}$ outputs whatever $\mathcal{A}_{sbbcfa}$ outputs. Note that $\mathcal{A}_{cpa}$ provides $\mathcal{A}_{sbbcfa}$ with a perfect simulation of the auxiliary experiment. It then follows that:

$$\Pr\left[\,W^A\,\right] = \Pr\left[\, d \leftarrow_{\$} \{0,1\} \, : \, \mathbf{Exp}^{\text{ind-cpa-d}}_{\mathcal{SE}}(\mathcal{A}_{cpa}) = d \,\right] . \tag{6.27}$$

Combining equations (6.26) and (6.27) yields (6.23), as desired.

□

6.7 Summary

The SSH attack of [1] and the IPsec attacks of [30] serve to show that, contrary to common belief, ciphertext fragmentation cannot always be abstracted out from security models. In this chapter, we have initiated the formal study of symmetric encryption in the presence of ciphertext fragmentation. In many practical settings, such as that of secure protocols operating over TCP/IP, the underlying channel is the one that is implicit in our models. That is, the channel that allows adversarially-controlled fragmentation, but which preserves the order of the fragments in the absence of an adversary. In addition to making correctness, confidentiality, and boundary hiding more challenging, ciphertext fragmentation introduces new security concerns such as the DoS attacks which we described. Thus, in addition to narrowing the gap between theory and practice, we find that the study of ciphertext fragmentation to be of interest in its own right. We now conclude with a brief comparison of our constructions and the SSH variants. This is shown in Table 6.1, which lists the security notions that are met by each construction. The last column indicates which schemes are n-DOS-sfCFA or n-DOS-sbbCFA secure for a value of n strictly smaller than the maximum message length supported by the scheme. We reiterate that stateful InterMAC is the only scheme to achieve all four security notions simultaneously. As explained in Section 6.4.2 we do not expect the SBB variant of InterMAC to be BH-sbbCFA secure. However it is BH-CPA secure if the underlying scheme is IND\$-CPA secure and the underlying MAC is pseudorandom. Although we did not prove this explicitly it follows easily by first showing that $\mathcal{IM}^*$ is IND\$-CPA secure and then applying Theorem 6.2.

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

Table 6.1: Security comparison of encryption schemes supporting fragmentation.

Bibliography

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

[2] Nadhem J. AlFardan and Kenneth G. Paterson. Plaintext-recovery attacks against datagram TLS. In *Network and Distributed System Security Symposium (NDSS 2012)*, 2012.

[3] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *IEEE Symposium on Security and Privacy*, pages 526–540, 2013.

[4] R. Atkinson. IP encapsulating security payload (ESP). RFC 1827 (Proposed Standard), August 1995. Obsoleted by RFC 2406.

[5] Gregory V. Bard. A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL. In *SECRYPT*, pages 99–109, 2006.

[6] Gregory V. Bard. Blockwise-adaptive chosen-plaintext attack and online modes of encryption. In *IMA Int. Conf.*, pages 129–151, 2007.

[7] Aurélie Bauer, Jean-Sébastien Coron, David Naccache, Mehdi Tibouchi, and Damien Vergnaud. On the broadcast and validity-checking security of PKCS#1 v1.5 encryption. In *ACNS*, pages 1–18, 2010.

[8] M. Bellare, T. Kohno, and C. Namprempre. The secure shell (SSH) transport layer encryption modes. RFC 4344 (Proposed Standard), January 2006.

[9] Mihir Bellare. Practice-oriented provable security. In *Lectures on Data Security*, pages 1–15, 1998.

[10] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *CRYPTO*, pages 602–619, 2006.

[11] Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online ciphers and the Hash-CBC construction. In *CRYPTO*, pages 292–309, 2001.

[12] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403, 1997.

[13] Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. *IACR Cryptology ePrint Archive*, 2004:309, 2004.

[14] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *CRYPTO*, pages 15–28, 1995.

[15] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In *CRYPTO*, pages 341–358, 1994.

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

[17] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, pages 531–545, 2000.

[18] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO*, pages 232–249, 1993.

[19] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[20] Steven M. Bellovin. Problem areas for the IP security protocols. In *in Proceedings of the Sixth Usenix Unix Security Symposium*, pages 205–214, 1996.

[21] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *CRYPTO*, pages 1–12, 1998.

[22] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. Security of symmetric encryption in the presence of ciphertext fragmentation. In *EUROCRYPT*, pages 682–699, 2012.

[23] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. On symmetric encryption with distinguishable decryption failures. In *FSE*, 2013.

[24] Alexandra Boldyreva and Nut Taesombut. Online encryption schemes: New security notions and constructions. In *CT-RSA*, pages 1–14, 2004.

[25] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: the insecurity of 802.11. In *MOBICOM*, pages 180–189, 2001.

[26] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[27] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In *CRYPTO*, pages 583–599, 2003.

[28] Douglas Comer, editor. *Internetworking with TCP/IP - Principles, Protocols, and Architectures, Fourth Edition.* Prentice-Hall, 2000.

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

[30] Jean Paul Degabriele and Kenneth G. Paterson. On the (in)security of IPsec in MAC-then-encrypt configurations. In *ACM Conference on Computer and Communications Security*, pages 493–504, 2010.

[31] Jean Paul Degabriele, Kenneth G. Paterson, and Gaven J. Watson. Provable security in the real world. *IEEE Security & Privacy*, 9(3):33–41, 2011.

[32] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176.

[33] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746, 6176.

[34] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[35] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[36] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In *EUROCRYPT*, pages 355–374, 2012.

[37] Thai Duong and Juliano Rizzo. Cryptography in the web: The case of cryptographic design flaws in ASP.NET. In *IEEE Symposium on Security and Privacy*, pages 481–489, 2011.

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

[39] Niels Ferguson and Bruce Schneier. A cryptographic evaluation of IPsec. *Counterpane Internet Security, Inc*, 3031, 2000.

[40] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering - Design Principles and Practical Applications.* Wiley, 2010.

[41] Pierre-Alain Fouque, Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Authenticated on-line encryption. In *Selected Areas in Cryptography*, pages 145–159, 2003.

[42] Pierre-Alain Fouque, Antoine Joux, and Guillaume Poupard. Blockwise adversarial model for on-line ciphers and symmetric encryption schemes. In *Selected Areas in Cryptography*, pages 212–226, 2004.

[43] Pierre-Alain Fouque, Gwenaëlle Martinet, and Guillaume Poupard. Practical symmetric on-line encryption. In *FSE*, pages 362–375, 2003.

[44] S. Frankel, R. Glenn, and S. Kelly. The AES-CBC cipher algorithm and its use with IPsec. RFC 3602 (Proposed Standard), September 2003.

[45] S. Frankel and H. Herbert. The AES-XCBC-MAC-96 algorithm and its use with IPsec. RFC 3566 (Proposed Standard), September 2003.

[46] R. Glenn and S. Kent. The NULL encryption algorithm and its use with IPsec. RFC 2410 (Proposed Standard), November 1998.

[47] Oded Goldreich. On post-modern cryptography. *IACR Cryptology ePrint Archive*, 2006:461, 2006.

[48] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479, 1984.

[49] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *CRYPTO*, pages 276–288, 1984.

[50] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377, 1982.

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

[52] Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction attacks against several public-key cryptosystems. In *ICICS*, pages 2–12, 1999.

[53] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[54] R. Housley. Using advanced encryption standard (AES) counter mode with IPsec encapsulating security payload (ESP). RFC 3686 (Proposed Standard), January 2004.

[55] R. Housley. Using advanced encryption standard (AES) CCM mode with IPsec encapsulating security payload (ESP). RFC 4309 (Proposed Standard), December 2005.

[56] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *FOCS*, pages 230–235, 1989.

[57] Tibor Jager and Juraj Somorovsky. How to break XML encryption. In *ACM Conference on Computer and Communications Security*, pages 413–422, 2011.

[58] Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Blockwise-adaptive attackers: Revisiting the (in)security of some provably secure encryption models: CBC, GEM, IACBC. In *CRYPTO*, pages 17–30, 2002.

[59] Charanjit S. Jutla. Encryption modes with almost free message integrity. In *EUROCRYPT*, pages 529–544, 2001.

[60] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), December 2005. Obsoleted by RFC 5996, updated by RFC 5282.

[61] S. Kent. IP authentication header. RFC 4302 (Proposed Standard), December 2005.

[62] S. Kent. IP encapsulating security payload (ESP). RFC 4303 (Proposed Standard), December 2005.

[63] S. Kent and R. Atkinson. IP encapsulating security payload (ESP). RFC 2406 (Proposed Standard), November 1998. Obsoleted by RFCs 4303, 4305.

[64] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), November 1998. Obsoleted by RFC 4301, updated by RFC 3168.

[65] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005. Updated by RFC 6040.

[66] A. Keromytis and N. Provos. The use of HMAC-RIPEMD-160-96 within ESP and AH. RFC 2857 (Proposed Standard), June 2000.

[67] Neal Koblitz and Alfred Menezes. Another look at ”provable security”. *J. Cryptology*, 20(1):3–37, 2007.

[68] Neal Koblitz and Alfred Menezes. Another look at security definitions. *IACR Cryptology ePrint Archive*, 2011:343, 2011.

[69] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In *CRYPTO*, pages 310–331, 2001.

[70] C. Madson and R. Glenn. The use of HMAC-SHA-1-96 within ESP and AH. RFC 2404 (Proposed Standard), November 1998.

[71] James Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS #1 v2.0. In *CRYPTO*, pages 230–238, 2001.

[72] Ueli Maurer. Constructive cryptography - a new paradigm for security definitions and proofs. In *TOSCA*, pages 33–56, 2011.

[73] Ueli Maurer and Björn Tackmann. On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption. In *ACM Conference on Computer and Communications Security*, pages 505–515, 2010.

[74] D. McGrew and J. Viega. The use of galois message authentication code (GMAC) in IPsec ESP and AH. RFC 4543 (Proposed Standard), May 2006.

[75] David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In *INDOCRYPT*, pages 343–355, 2004.

[76] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113, 6649, 6806.

[77] Kenneth G. Paterson. A cryptographic tour of the IPsec standards. *IACR Cryptology ePrint Archive*, 2006:97, 2006.

[78] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In *ASIACRYPT*, pages 372–389, 2011.

[79] Kenneth G. Paterson and Gaven J. Watson. Plaintext-dependent decryption: A formal security treatment of SSH-CTR. In *EUROCRYPT*, pages 345–361, 2010.

[80] Kenneth G. Paterson and Gaven J. Watson. Authenticated-encryption with padding: A formal security treatment. In *Cryptography and Security*, pages 83–107, 2012.

[81] Kenneth G. Paterson and Arnold K. L. Yau. Cryptography in theory and practice: The case of encryption in IPsec. In *EUROCRYPT*, pages 12–29, 2006.

[82] R. Pereira and R. Adams. The ESP CBC-mode cipher algorithms. RFC 2451 (Proposed Standard), November 1998.

[83] E. Rescorla and N. Modadugu. Datagram transport layer security version 1.2. RFC 6347 (Proposed Standard), January 2012.

[84] Phillip Rogaway. Problems with proposed IP cryptography. Unpublished manuscript, April 1995.

[85] Phillip Rogaway. Nonce-based symmetric encryption. In *FSE*, pages 348–359, 2004.

[86] Phillip Rogaway. On the role of definitions in and beyond cryptography. In *ASIAN*, pages 13–32, 2004.

[87] Phillip Rogaway. Practice-oriented provable security and the social construction of cryptography. *Unpublished essay based on an invited talk at Eurocrypt*, 2009.

[88] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security*, pages 196–205, 2001.

[89] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[90] Tom Shrimpton. A characterization of authenticated-encryption as a form of chosen-ciphertext security. *IACR Cryptology ePrint Archive*, 2004:272, 2004.

[91] JH. Song, R. Poovendran, J. Lee, and T. Iwata. The AES-CMAC algorithm. RFC 4493 (Informational), June 2006.

[92] William Stallings. *Network Security Essentials - Applications and Standards (4. ed., internat. ed.).* Pearson Education, 2010.

[93] Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS ... In *EUROCRYPT*, pages 534–546, 2002.

[94] J. Viega and D. McGrew. The use of galois/counter mode (GCM) in IPsec encapsulating security payload (ESP). RFC 4106 (Proposed Standard), June 2005.

[95] David Wagner and Bruce Schneier. Analysis of the SSL 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, pages 29–40, 1996.

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

[97] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610 (Informational), September 2003.

[98] T. Ylonen and C. Lonvick. The secure shell (SSH) authentication protocol. RFC 4252 (Proposed Standard), January 2006.

[99] T. Ylonen and C. Lonvick. The secure shell (SSH) connection protocol. RFC 4254 (Proposed Standard), January 2006.

[100] T. Ylonen and C. Lonvick. The secure shell (SSH) protocol architecture. RFC 4251 (Proposed Standard), January 2006.

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