Diss. ETH No. 22140

A Theory of Secure Communication

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

Björn Tackmann
Dipl. Math. Dipl. Inform., Universität Karlsruhe (TH)

born on April 13, 1980
citizen of the Federal Republic of Germany

accepted on the recommendation of

Prof. Dr. Ueli Maurer, examiner
Prof. Dr. Mihir Bellare, co-examiner
Prof. Dr. Adrian Perrig, co-examiner

2014

Dedicated to my dear mother

RENATE TACKMANN
* 1948 † 2013

Alles im Leben hat seine Zeit.

Acknowledgments

First and foremost, I would like to thank my advisor Ueli Maurer for giving me the opportunity to do a PhD in his group, and for his continuous advice and support as well as the numerous inspiring and joyful discussions we had. With endless patience and confidence, and with his unique ability of explication, he taught me to find and tackle fundamental research questions. I know that I could not have made this unique experience elsewhere. Also beyond research, Ueli's door has always been open for me. I am particularly grateful for his unconditional backing during the difficult period in my family, as well as for his guidance and support in all professional matters. I was lucky to find an advisor who not only guided me through my studies and research, but also backed and encouraged me in all my endeavors, no matter if professional or private, and has since become an influential person in my life.

I want to thank Mihir Bellare and Adrian Perrig for serving on my examination committee and for providing valuable feedback and being available for fruitful discussions.

During my PhD studies, I had the fortunate opportunity to discuss and work with several outstanding researchers and students. I especially want to thank my co-authors Andreas Rüedlinger, Cristina Onete, Daniele Venturi, Grégory Demay, Jonathan Katz, Juan Garay, Markulf Kohlweiss, Peter Gaži, Sandro Coretti, and Vassilis Zikas. Additionally, I want to thank Anja Lehmann, Cas Cremers, Christian Badertscher, Christian Matt, Christopher Portmann, Daniel Jost, Dennis Hofheinz, Jörn Müller-Quade, Joël Alwen, Kenny Paterson, Mario Strefler, Martin Hirt, Mihir Bellare, Phil Rogaway, Ran Canetti, Simon Knellwolf, Stefano Tessaro, Steven Myers, and Thomas Holenstein for interesting discussions and/or valuable feedback they provided for my work. I am particularly thankful to Christoph Lucas, who was my long-term office mate and sparring partner and became a very dear friend of mine. I would also like to thank our secretary Beate Bernhard who has supported me in all administrative tasks, and Cristina and Gunnar who proof-read earlier versions of my thesis and provided helpful feedback.

I would like to express my deep gratitude to my family for continuously giving me love, encouragement, and support.

My life at ETH and more broadly in Zurich has been made enjoyable by the many wonderful people I've met here. It is infeasible to list them all, but besides my colleagues at ETH I'd like to thank Anja, Armin, Chrissy, Florian & Carmen, Franziska, Raffi, Robin, Roger, Silvia & Igor, Simon, Stefan, Thomas, Urs, and especially Elke for bringing sunshine to my life.

Finally, I would like to thank the Swiss National Science Foundation for partially funding my position as a part of project no. 200020-132794.

Abstract

Cryptographic schemes and protocols are essential tools for the protection of communications and IT applications. One of their currently most widespread uses is in securing the communication over networks of electronic devices, primarily the Internet, where cryptographic protocols are employed in day-to-day services such as online banking as well as for guaranteeing the privacy of personal (electronic) communication.

Since the security of a cryptographic protocol or scheme, in contrast to its correctness, cannot simply be demonstrated, the now widely accepted paradigm is that one builds confidence in cryptographic protocols by providing a (mathematical) proof of their security. Such an approach first requires one to define what it means for a cryptographic protocol to be secure. Then, one specifies the assumptions that one is willing to make. This includes assumptions about the resources available to the protocol like communication channels or memory; it also comprises assumed restrictions of the attacker such as its inability to access the protocol memory or to solve certain seemingly difficult computational problems. Finally, one provides a formal proof that the validity of the assumptions implies the security of the protocol.

The approach of proving the security of cryptographic schemes can be traced back to Shannon's 1949 work about one-time pad encryption. Yet, his tailor-made and purely information-theoretic definition and proof do not generalize to other types of schemes. In 1976, Diffie and Hellman introduced a cryptographic (public-key) scheme whose security was explicitly based on the conjectured hardness of a specific computational problem, but they did not formally define or prove security. For the case of public-key encryption, a security definition was introduced by Goldwasser and Micali in 1984. Even today, the most widely used definitions in the context of secure communication follow the paradigm introduced in their work, which formalizes the attacker's inability to perform certain attacks on the scheme. This approach has several drawbacks since it does not make the guarantees that a scheme provides in a specific application explicit, and more importantly it does not

support a modularization of schemes or proofs. Support for such a modularization was then achieved by the general frameworks introduced by Canetti and, independently, by Pfitzmann and Waidner in 2001. Since these models are described as generalizations of earlier ones, they still carry legacy formalism inherited from the previous definitions, and additionally introduced artifacts through design choices made during the generalization. In 2011, Maurer and Renner proposed that instead of extending and generalizing existing definitions, which inherently leads to a more and more complex formalism, a theory of cryptography should follow an axiomatic perspective. This approach, in which one initially defines the considered concepts at an abstract level and iteratively refines them whenever necessary, assures that the obtained definitions are clean and simple, and exactly capture the desired concepts.

This thesis describes a coherent theory of secure communication following the described axiomatic approach. The first contribution is the specification of a formal framework that instantiates the concepts of abstract and constructive cryptography (Maurer and Renner, ICS 2011) and in which security statements about concrete cryptographic protocols can be phrased and proven. The second contribution is the application of this framework in the setting of secure communication. This comprises the proof of well-known protocols with respect to the new definitions, the comparison to several existing security notions from the literature, and the analysis of a new unilateral key-establishment protocol. The third contribution is a modular proof of the "record layer" of the Transport Layer Security (TLS) protocol, the currently most widely deployed cryptographic protocol on the Internet. The contributions are described in more detail in the following paragraphs.

A framework for constructive cryptography. The most fundamental question for a security definition is *what it actually means* for a protocol to be secure. The constructive cryptography paradigm follows the idea that the requirements are ultimately mandated by the application: given the *assumed resources* that are available to the protocol, such as an insecure communication network and a public-key infrastructure, the goal of a cryptographic protocol is to *construct*, in a well-defined sense, a *desired resource* that the parties executing the protocol wish to obtain, such as a secure end-to-end communication channel. One important property of such *construction statements* is that they can be chained: a constructed resource can be used as an assumed resource in a subsequent construction step. This allows to build complex protocols as a sequence of simple construction steps, which results in a modular proof (since each construction statement can be proven in isolation) and a modular structure (because each construction step can be replaced individu-

ally by an alternative one or re-used in other protocols).

While the construction concept is universal and independent of cryptography, the definition of what it means for a protocol to actually *achieve* a construction is specific to each considered setting, that is, the number and type of parties that are involved. Following the work of Maurer and Renner, the setting of secure communication—which involves two or more honest parties and one potential attacker that attempts to eavesdrop on or to disturb the communication—results in a so-called simulation-based security definition with a single simulator. The concept of a simulator, albeit in a slightly different formal sense, was introduced by Goldwasser, Micali, and Rackoff in 1985 and also appears as a crucial ingredient in the general frameworks of Canetti and of Pfitzmann and Waidner.

This thesis instantiates the concepts of abstract and constructive cryptography to allow for a formal analysis of concrete cryptographic protocols. The abstract system concept used by Maurer and Renner to describe the construction notion is formalized and instantiated in terms of *discrete systems*, i.e., systems that communicate by sending and receiving messages and which are described by their input/output behavior. We then define a construction notion based on Maurer's abstract reduction theory and show that the notion is indeed composable. Additionally, we show how *parametrized* construction statements are defined. These statements model settings where the security bounds that can be shown for a protocol depend on parameters which might be static (like the key length used in an encryption scheme) or even *a priori* unbounded and only determined while the analyzed protocol is used (like the length or number of encrypted messages). Overall, we develop a formal framework in which a broad class of cryptographic protocols can be analyzed, which is precisely defined and inherits the main advantages of constructive cryptography like the clear semantics of the security statements and the inherent modularity.

Constructing resources for secure communication. The capability of two or more parties to communicate can be formalized as a *channel* that is available to these parties. In the setting of secure communication where a potential attacker may attempt to eavesdrop or to disturb the communication, one can describe different levels of security by explicitly specifying to which extent the attacker can access the communication. This results in resources such as *insecure channels* where the attacker can completely control the communication, *authenticated channels* where only the legitimate sender can send messages but the attacker can still eavesdrop on the communication, and *secure channels* in which both the authenticity and the confidentiality of the communication are guaranteed. A further important type of resource models the concept

of shared (secret) randomness such as different types of *cryptographic keys* with specific security properties.

Each of the described resources can appear either as an assumed or as a constructed resource in a construction statement. Assumed resources can either be assumed to exist in the real world (this is the case for insecure communication channels, which model communication via the Internet), or they can be constructed by other protocols from other assumed resources; the composability of constructions guarantees that using constructed resources in a subsequent construction is sound. For example, a message authentication code (MAC) constructs an authenticated channel from an insecure channel and a shared cryptographic key, and a symmetric encryption scheme constructs a fully secure channel from an additional cryptographic key and an authenticated channel, which could be the one constructed by the MAC. A key-establishment protocol, such as Diffie-Hellman, constructs from a pair of authenticated channels a cryptographic key.

The thesis formalizes resources such as the ones listed above and proves the validity of various construction steps. Beyond the constructions achieved by MACs and symmetric encryption, we show how one models anonymity following the constructive paradigm and prove that a key-private (and robust) public-key encryption scheme constructs a receiver-anonymous confidential channel from appropriate resources. Finally, we describe a unilateral key-establishment protocol and prove that it constructs a unilaterally authenticated key from insecure and authenticated communication channels. Beyond proving concrete schemes, we also show which construction statements are implied by several existing security notions from the literature.

A constructive perspective on the TLS record-layer protocol. Besides providing sound definitions and designing secure cryptographic protocols, a further task of practice-oriented cryptography is to analyze the security of existing cryptographic protocols. This task is particularly difficult if the considered protocol has not been designed with provable security in mind. Complex cryptographic protocols (like for example SSL/TLS, IPsec, or ssh) are often designed in a somewhat ad-hoc fashion, without a security proof. In fact, a substantial number of practical cryptographic protocols have been broken in the past, including all the three protocols mentioned above. The exploited vulnerabilities are rarely caused by a failure of the underlying primitives, but mostly occur because the different elements of the protocols are in subtle ways not compatible.

The TLS protocol is the most widely deployed cryptographic protocol on the Internet. Initially developed by Netscape as Secure Socket Layer (SSL) for securing the transmission between web servers and web browsers, it has

been adopted to many other scenarios such as securing connections to mail, directory, or database servers, and even some virtual private networks are based on (parts of) the protocol.

After various vulnerabilities have been found in the protocol and patched in subsequent releases, the most recent version TLS 1.2 appears to achieve a reasonable level of security. Indeed, a long line of papers in the cryptographic literature focuses on proving the security of (parts of) the TLS protocol. In this thesis, we prove the security of the *record layer* of the TLS protocol as a construction statement. The task of this sub-protocol is to actually protect payload messages after a cryptographic key has been established in the so-called *handshake*, another sub-protocol. Our result shows that the use of the particular schemes applied in TLS 1.2 is sound. The composability of the security definition ensures that the analysis of the record-layer protocol can be combined with an analogous analysis of the handshake protocol.

Zusammenfassung

Kryptographische Verfahren und Protokolle sind zentrale Werkzeuge für den Schutz von Kommunikation und IT-Anwendungen. Eine der gebräuchlichsten Verwendungen ist der Schutz der Kommunikation in Netzwerken elektronischer Geräte, insbesondere dem Internet, wo kryptographische Protokolle essentiell sind für die Sicherheit verbreiteter Anwendungen wie Online Banking ebenso wie für den Schutz von privater Kommunikation.

Da die Sicherheit eines kryptographischen Protokolls, im Gegensatz zu seiner Funktionalität, nicht einfach demonstriert werden kann, hat sich die Ansicht durchgesetzt, dass das Vertrauen in kryptographische Protokolle durch einen (mathematischen) Beweis ihrer Sicherheit gestützt werden soll. Dieser Ansatz erfordert zunächst die Entwicklung einer formalen Definition für die Sicherheit eines Protokolls. Daraufhin werden die Annahmen, unter denen das Protokoll sicher sein soll, spezifiziert. Diese Annahmen umfassen zum Einen die Ressourcen, die dem Protokoll zur Verfügung stehen, etwa Kommunikationskanäle oder Speicher. Zum Anderen enthalten sie vermutete Beschränkungen möglicher Angreifer, etwa dass diese keinen Zugriff auf den Speicher des Protokolls haben, oder dass sie ausserstande sind, gewisse als schwierig erachtete mathematische Probleme zu lösen. Schlussendlich wird gezeigt, dass die Sicherheit des Protokolls aus der Gültigkeit der Annahmen folgt.

Der Ansatz, die Sicherheit kryptographischer Verfahren zu beweisen, geht zurück auf Shannons Arbeit aus dem Jahr 1949 über die One-Time-Pad-Verschlüsselung. Seine Definition und sein Beweis sind jedoch spezifisch für das betrachtete Verfahren und lassen sich nicht direkt auf andere Fälle übertragen. Im Jahr 1976 schlugen Diffie und Hellman ein asymmetrisches kryptographisches Verfahren vor, das explizit auf der vermuteten Schwierigkeit eines bestimmten mathematischen Problems beruht. Ihre Arbeit enthält jedoch keine formale Definition für die Sicherheit des Verfahrens. Für den Fall der asymmetrischen Verschlüsselung wurde eine solche Definition im Jahr 1984 von Goldwasser und Micali vorgeschlagen. Viele der heute verbreiteten Defi-

nitionen im Bereich der Kommunikationssicherheit basieren auf ihrem Ansatz, der die Unmöglichkeit bestimmter Klassen von Angriffen auf das Verfahren formalisiert. Dieser Ansatz hat einige Nachteile, weil er die Garantien, die ein Verfahren in einer bestimmten Anwendung erreicht, nicht explizit macht. Insbesondere ermöglicht er nicht die Modularisierung komplexer Verfahren oder ihrer Beweise. Eine solche Modularisierung wird von den allgemeinen Modellen unterstützt, die Canetti und unabhängig Pfitzmann und Waidner im Jahr 2001 einführten. Da diese Modelle jedoch als Verallgemeinerungen aus den vorhergehenden Definitionen hervorgegingen, basieren sie noch immer auf einem ähnlichen Formalismus und fügten zudem im Zuge der Verallgemeinerung weitere Artefakte hinzu. Im Jahr 2011 schlugen Maurer und Renner vor, dass eine Theorie der Kryptographie einem axiomatischen Ansatz folgen sollte, anstelle existierende Definitionen zu erweitern und zu verallgemeinern. Dieser Ansatz, in dem man zu Beginn die betrachteten Konzepte auf einem abstrakten Niveau beschreibt und dann iterativ verfeinert, stellt sicher, dass die daraus hervorgehenden Definitionen sauber und einfach sind, und genau die gewünschten Konzepte beschreiben.

Diese Dissertation folgt dem beschriebenen axiomatischen Ansatz und beschreibt eine kohärente Theorie der Kommunikationssicherheit. Der erste Beitrag ist die Spezifikation eines formalen Modells, das den Konzepten der abstrakten und konstruktiven Kryptographie (Maurer und Renner, ICS 2011) folgt, und in dem Aussagen über die Sicherheit konkreter kryptographischer Protokolle beschrieben und bewiesen werden können. Der zweite Beitrag ist die Anwendung dieses Modells auf den Bereich der Kommunikationssicherheit. Dies umfasst den Beweis verbreiteter Protokolle relativ zu den neuen Definitionen, den Vergleich mit verschiedenen existierenden Definitionen aus der Literatur, und die Analyse eines neuen einseitig authentifizierten Schlüsselaustauschprotokolls. Der dritte Beitrag ist ein modularer Beweis des „Record Layer“-Teilprotokolls des „Transport Layer Security (TLS)“-Protokolls, des momentan im Internet am häufigsten verwendeten kryptographischen Protokolls. Die Beiträge werden in den folgenden Absätzen genauer beschrieben.

Ein Modell für konstruktive Kryptographie. Die grundlegende Frage für eine Sicherheitsdefinition ist, was *Sicherheit* eines Protokolls genau bedeutet. Das Paradigma der konstruktiven Kryptographie folgt der Idee, dass die Anforderungen von der Anwendung vorgegeben werden: gegeben die *angenommenen Ressourcen*, die dem Protokoll zur Verfügung stehen (wie ein unsicheres Kommunikationsnetzwerk und eine Public-Key Infrastruktur), ist das Ziel eines kryptographischen Protokolls, eine *gewünschte Ressource* (wie einen sicheren Ende-zu-Ende Kommunikationskanal) in einem wohldefinier-

ten Sinn zu *konstruieren*. Eine wichtige Eigenschaft solcher *Konstruktionsaussagen* ist, dass sie verkettet werden können: Eine konstruierte Ressource kann as angenommene Ressource in einem folgenden Konstruktionsschritt verwendet werden. Diese Struktur ermöglicht den Entwurf komplexer Protokolle als eine Folge einfacher Konstruktionsschritte, was sowohl in einem modularen Beweis (weil jede Konstruktionsaussage einzeln bewiesen werden kann) als auch in einer modularen Protokollstruktur (weil jeder Konstruktionsschritt einzeln durch einen anderen Schritt ersetzt oder in anderen Protokollen verwendet werden kann) resultiert.

Während das Konzept der Konstruktionsaussagen universell und unabhängig von der Anwendung in der Kryptographie ist, ist die genaue Instantiierung spezifisch für den betrachteten Anwendungsfall, genauer gesagt die Anzahl und die Art der involvierten Parteien. Der Anwendungsfall der Kommunikationssicherheit umfasst zwei oder mehr ehrliche Parteien sowie möglicherweise einen Angreifer, der versucht, die Kommunikation zu belauschen oder zu stören. Nach Maurer und Renner führt dieser Anwendungsfall zu einer sogenannten simulationsbasierten Definition mit einem einzelnen Simulator. Das Konzept des Simulators, obschon in einer leicht anderen Formalisierung, wurde im Jahr 1985 von Goldwasser, Micali und Rackoff eingeführt und erscheint auch als wichtiger Teil der allgemeinen Modelle von Canetti und von Pfitzmann und Waidner.

Diese Dissertation instiantiiert die Konzepte der abstrakten und konstruktiven Kryptographie, um die formale Analyse konkreter kryptographischer Protokolle zu ermöglichen. Das Konzept der abstrakten Systeme, das von Maurer und Renner zur Beschreibung des Konstruktionsbegriffs verwendet wurde, wird formalisiert und im Sinne von diskreten Systemen, also Systemen, die durch das Senden und Empfangen von Nachrichten kommunizieren, instantiiert. Dann beschreiben wir einen Konstruktionsbegriff basierend auf Maurers abstrakter Reduktionstheorie und zeigen dass diese Konstruktionen tatsächlich verkettet werden können. Zudem zeigen wir eine Erweiterung des Konstruktionsbegriffs auf parametrisierte Aussagen. Solche Aussagen modellieren Situationen in denen die Schranken, die für ein Protokoll gezeigt werden können, von Parametern abhängen die statisch sein können (wie die Länge der Schlüssel, die in einem Verschlüsselungsverfahren verwendet werden), oder zunächst unbeschränkt und abhängig von der konkreten Verwendung des Verfahrens (wie die Länge oder Anzahl der verschlüsselten Nachrichten). Insgesamt entwickeln wir ein formales Modell, in dem eine grosse Klasse kryptographischer Protokolle analysiert werden kann. Das Modell ist präzise definiert und hat die Vorteile konstruktiver Kryptographie wie die klare Semantik der Sicherheitsaussagen und die inhärente Modularität.

Konstruktion von Ressourcen für Kommunikationssicherheit. Die Fähigkeit von zwei oder mehr Parteien zu kommunizieren kann als ein *Kanal*, der den Parteien zur Verfügung steht, formalisiert werden. Im Anwendungsfall der Kommunikationssicherheit, in der ein potenzieller Angreifer möglicherweise versucht die Kommunikation zu belauschen oder zu stören, können verschiedene Sicherheitsniveaus beschrieben werden indem der mögliche Zugriff des Angreifers auf die Kommunikation explizit spezifiziert wird. Dieses Vorgehen resultiert in Ressourcen wie *unsicheren Kanälen*, bei denen der Angreifer die Kommunikation vollständig kontrolliert, *authentifizierten Kanälen*, bei denen nur der legitime Sender Nachrichten senden kann, es dem Angreifer jedoch immer noch möglich ist, die Kommunikation zu belauschen, oder *sicheren Kanälen* in denen sowohl die Authentizität als auch die Geheimhaltung der Nachrichten garantiert wird. Eine weitere wichtige Art von Ressourcen modelliert das Konzept von gemeinsamen (geheimen) Zufallswerten wie unterschiedlichen Arten kryptographischer Schlüssel mit spezifischen Sicherheitseigenschaften.

Jede der beschriebenen Ressourcen kann entweder als eine angenommene oder als eine konstruierte Ressource in einer Konstruktionsaussage erscheinen. Angenommene Ressourcen können entweder als in der realen Welt existierend angenommen werden (dies ist der Fall für unsichere Kommunikationskanäle, die die Kommunikation über das Internet modellieren), oder sie können von anderen Protokollen von weiteren Ressourcen konstruiert werden. Die Möglichkeit der Verkettung von Konstruktionsaussagen garantiert, dass die Verwendung von konstruierten Ressourcen in folgenden Konstruktionsaussagen möglich ist. Zum Beispiel konstruiert ein Nachrichtenauthentifizierungscode (MAC) einen authentifizierten Kanal aus einem unsicheren Kanal und einem gemeinsamen kryptographischen Schlüssel. Ein (symmetrisches) Verschlüsselungsverfahren konstruiert einen sicheren Kanal aus einem weiteren gemeinsamen Schlüssel und einem authentifizierten Kanal, der zum Beispiel der durch den MAC konstruierte Kanal sein könnte. Ein Schlüsselaustauschprotokoll, etwa Diffie-Hellman, konstruiert aus einem Paar authentifizierter Kanäle einen kryptographischen Schlüssel.

Diese Dissertation formalisiert Ressourcen wie die oben beschriebenen und beweist die Gültigkeit verschiedener Konstruktionsaussagen. Ausser den Aussagen für MACs und symmetrische Verschlüsselung zeigen wir auch, wie Anonymität im Sinne des konstruktiven Paradigmas modelliert wird, und beweisen, dass eine bestimmte Art von asymmetrischen Verschlüsselungsverfahren einen geheimen Kanal, der zudem die Anonymität des Empfängers garantiert, aus passenden Ressourcen konstruiert. Zuletzt beschreiben wir ein einseitig authentifiziertes Schlüsselaustauschprotokoll und beweisen, dass es einen einseitig authentifizierten Schlüssel aus einem unsicheren und einem

authentifizierten Schlüssel konstruiert. Ausser dem Beweis konkreter Verfahren zeigen wir auch die Beziehung unserer Definitionen zu verschiedenen existierenden Definitionen aus der Literatur und beschreiben, welche Konstruktionsaussagen von diesen Definitionen impliziert werden.

Eine konstruktive Analyse des TLS-„Record Layer“-Teilprotokolls. Ausser der Entwicklung von Sicherheitsdefinitionen und dem Entwurf neuer sicherer kryptographischer Protokolle ist eine weitere Aufgabe praxisorientierter Kryptographie die Analyse der Sicherheit von existierenden kryptographischen Protokollen. Diese Aufgabe ist besonders schwierig, wenn die betrachteten Protokolle nicht mit Blick auf beweisbare Sicherheit entworfen wurde. Komplexe kryptographische Protokolle (wie etwa SSL/TLS, IPsec oder ssh) werden häufig inkonsistent und ohne einen Sicherheitsbeweis entwickelt. Tatsächlich wurde ein beträchtlicher Anteil verbreiteter kryptographischer Protokolle in der Vergangenheit gebrochen, inklusive aller drei oben genannter Protokolle. Die Angriffe zielen selten auf das Versagen zugrundeliegender kryptographischer Primitiven, sondern treten meistens auf, weil verschiedene Elemente der Protokolle auf subtile Art inkompatibel sind.

Das TLS-Protokoll ist das am weitesten verbreitete kryptographische Protokoll im Internet. Es wurde ursprünglich von Netscape als „Secure Socket Layer (SSL)“ für die Absicherung der Datenübertragung zwischen Web-Servern und Web-Browsern entwickelt, wurde aber mittlerweile auf weitere Anwendungsfälle wie die Absicherung der Verbindungen zu Mail-Servern, Verzeichnisdiensten oder Datenbanken. Auch einige virtuelle private Netzwerke (VPN) basieren auf Teilen des Protokolls.

Nachdem mehrere Angriffspunkte im Protokoll gefunden und in folgenden Versionen behoben wurden, scheint die aktuelle Protokollversion TLS 1.2 ein angemessenes Sicherheitsniveau zu erreichen. Eine lange Folge von Arbeiten in der kryptographischen Literatur befasst sich mit dem Beweis von Teilen des TLS-Protokolls. In dieser Arbeit beweisen wir die Sicherheit des „Record Layer“-Teilprotokolls von TLS als Konstruktionsaussage. Dieses Teilprotokoll hat die Aufgabe, die übertragenen Anwendungsdaten zu schützen, nachdem ein gemeinsamer geheimer Schlüssel von einem anderen Teilprotokoll ausgehandelt wurde. Der Beweis zeigt, dass die Verwendung der Verfahren in TLS 1.2 tatsächlich sicher ist. Die Möglichkeit zur Verkettung der Sicherheitsdefinitionen stellt sicher, dass diese Analyse des Teilprotokolls mit einer entsprechenden Analyse des vorhergehenden Teilprotokolls kombiniert werden kann.

Contents

1 Introduction **1**
1.1 A Framework for Constructive Cryptography 3
1.1.1 The Construction Paradigm 3
1.1.2 Abstract Cryptography: The Axiomatic Formalization . . 4
1.1.3 Constructions for Secure Communication 6
1.1.4 Discrete Systems and Reductions 9
1.1.5 Related Work on Paradigms for Defining Security 11
1.1.6 Related Work on Models of Discrete Systems 14
1.2 Constructing Resources for Secure Communication 18
1.2.1 Secure Communication Between Two Parties 19
1.2.2 Receiver-Anonymous Communication 26
1.2.3 Unilaterally Authenticated Key Establishment 29
1.3 A Constructive Perspective on the TLS Record Layer 33
1.3.1 The TLS Protocol 33
1.3.2 Viewing the Record Layer Constructively 34
1.3.3 Related Work 35

2 Preliminaries **37**
2.1 Notation 37
2.1.1 General Notation 37
2.1.2 Sequences and Tuples 38
2.2 Abstract Reduction Theory 40
2.3 Random Systems 44
2.4 Cryptographic Schemes and Security Properties 50
2.4.1 Game-based Definitions 50
2.4.2 Message Authentication Codes 51
2.4.3 Symmetric Encryption 53
2.4.4 Public-Key Encryption 56
2.4.5 Key-Encapsulation Mechanisms 60

2.4.6 Signatures 61

3 A Framework for Constructive Cryptography 63
3.1 The Construction Concept 64
3.2 Abstract Systems 65
3.2.1 The Algebra of Abstract Systems 66
3.2.2 The Cryptographic Algebra 69
3.2.3 Instantiating the Operations of the Cryptographic Algebra 72
3.3 The Construction Notion 76
3.3.1 A Distinction Problem on Resources 77
3.3.2 The Notion of Construction 77
3.3.3 Parametrized Statements 83
3.4 Discrete Systems 84
3.4.1 The Algebra of Monotone Discrete Systems 85
3.4.2 Parametrized Statements and Uniformity 92
3.4.3 Discrete Games and Reductions 94
3.4.4 Specification of Discrete Systems 95

4 Resources for Secure Communication 99
4.1 Communication Channels 100
4.1.1 Formal Type and Availability Condition 100
4.1.2 Insecure Channels 101
4.1.3 Authenticated Channels 102
4.1.4 Confidential Channels 104
4.1.5 Secure Channels 106
4.1.6 Parametrized Channels 108
4.2 Keys and Shared-Randomness Resources 108
4.2.1 General Shared-Randomness Resources 109
4.2.2 Shared Secret Keys 109
4.2.3 Parametrized Randomness Resources 111
4.3 Message Authentication 112
4.3.1 Construction Based on a Shared URF 113
4.3.2 Construction Based on Weakly Unforgeable MACs 117
4.3.3 Constructing Ordered Authenticated Channels 119
4.4 Symmetric Encryption 121
4.4.1 The One-Time Pad and Stream Ciphers 121
4.4.2 CBC-Mode Encryption 128
4.4.3 Relation to Previous Security Notions 130
4.5 Combining Encryption and Authentication 140
4.5.1 Encrypt-then-Authenticate 140
4.5.2 Authenticate-then-Encrypt 141

4.5.3 Discussion 143
4.6 Receiver-Anonymous Communication 144
4.6.1 Resources for Receiver-Anonymous Communication . . . 144
4.6.2 Generic Construction using Public-Key Encryption 148
4.6.3 Achieving Confidential Receiver-Anonymous Communication 149
4.7 Unilateral Key Establishment 154
4.7.1 Constructing a Unilateral Key 155
4.7.2 Constructing the Authenticated Channel 157
4.7.3 Authenticating a Unilateral Key 163

5 The TLS Record Layer 167
5.1 Encryption 168
5.1.1 Cipher Suites Based on Stream Ciphers 168
5.1.2 Cipher Suites Based on CBC-Mode Encryption 172
5.2 Padding and Authentication 178
5.2.1 Construction Based on the XOR-Malleable Channel . . . 179
5.2.2 Construction Based on the Block-Malleable Channel . . 181

6 Conclusion 185

Chapter 1

Introduction

One of the most important and widely used applications of cryptographic techniques is the establishment of secure communication channels between two entities (e.g., a client and a server); many applications such as online banking, electronic mail, or remote file access require such a secure channel between the involved computers. Here, the term *secure* means that the communication does not cause harmful information to be leaked, and that the received messages originate from the assumed sender and are unmodified. In practice, however, the existing communication channels (such as communication over the Internet) are insecure and provide neither of these guarantees. Cryptographic schemes and protocols can be used to still achieve secure communication in this scenario.

The security of a protocol cannot be observed directly, because this would mean observing the *absence* of vulnerabilities to attacks. Even more, protocols need to be resilient against *any* type of attack, even those that are currently unknown and may only be discovered in the future. A key goal of research in cryptography is hence to *prove* the security of cryptographic protocols. This requires one to first provide a precise definition of what it means for a given protocol to be secure, then to identify and formalize the assumptions under which the protocol achieves the specified goal, and third to provide a formal proof that the validity of the assumptions implies the security of the protocol.

The described approach is surprisingly difficult to implement. One reason is that it is not even clear what "security" of a protocol means. An early and still widely used approach is to explicitly formalize classes of potential attacks on cryptographic schemes, and then prove the impossibility—or infeasibility—of these attacks. Yet, it is usually not clear whether a system composed of several cryptographic schemes, each with an individual attack-

based security proof, is secure for an overall attack-based security definition. In other words, these security definitions often do not *compose* in a meaningful way, impeding a modular design of security protocols. Another reason is that the capabilities of the adversary or attacker have to be specified in a way such that, on the one hand, the definitions are not overly strict and exclude schemes unnecessarily, but on the other hand, the model still captures all attacks possible in the real world. As most practical cryptographic schemes are secure only in a computational sense and could be broken by an attacker with unlimited computing power, security models must allow for an interpretation that reflects statements about computationally bounded attackers, which is usually achieved by modeling all entities such as the protocol and the attacker as Turing machines that run in polynomial time. Besides the limited applicability of asymptotic statements to real-world settings, a major criticism to this approach is that papers based on such models are generally not formulated in the "low-level" Turing machine formalism in which the models are specified. The papers often operate in a pseudo-code-type language without even specifying the exact objects, i.e. Turing machines, they consider and (claim to) prove statements about. Additionally, the environment in which cryptographic protocols are deployed is often complex and consists of multiple parties that interact in a distributed setting. A formal security model must be able to capture such interactions, while simultaneously allowing for a computational interpretation, and at the same time being simple enough to allow for comprehensible proofs. To date, no existing model satisfies all above requirements.

This thesis aims at filling the described gap. The first contribution is the specification of a formal framework in which security statements about cryptographic protocols can be precisely phrased and proven. The second contribution is the formalization of concrete security definitions and the analysis of protocols within the proposed framework. This includes the proof of several widely used schemes, a comparison of the obtained security definitions to several existing ones from the literature, and a new key-establishment protocol for the "unilateral" client-server scenario. The third contribution is the analysis of the "record layer" sub-protocol of TLS, the currently most widely used cryptographic protocol in the Internet.

Outline. The introduction begins in Section 1.1 with a high-level description of the formal framework. We describe the paradigm of constructive cryptography and sketch how this paradigm is instantiated for settings with a single external attacker; we then explain how we formalize cryptographic schemes and their applications. We discuss related work both with respect to other paradigms for security definitions (such as game-based definitions) in

Section 1.1.5 and with respect to other formal models for the instantiations (such as Turing machines or automata) in Section 1.1.6. This section of the introduction corresponds to Chapter 3 of the thesis.

In Section 1.2, we describe how the described framework is applied in the area of secure communication. We briefly discuss the security statements that are proven in Chapter 4, which are in the context of encryption and authentication (Section 1.2.1), anonymous communication (Section 1.2.2), and unilaterally authenticated key establishment (Section 1.2.3). The related literature for this part of the thesis is discussed in the respective sub-sections.

In Section 1.3, we describe how the TLS record-layer protocol can be analyzed from a constructive perspective. We also relate our results to other results in the literature. This section of the introduction corresponds to Chapter 5 of the thesis.

1.1 A Framework for Constructive Cryptography

The framework introduced in Chapter 3 follows the paradigm of constructive cryptography, which was introduced by Maurer and Renner [MR11]. Later, Maurer [Mau11] has exemplified the application of the paradigm using the one-time pad. These works, however, did not describe a formal framework in which concrete protocols can be formally described and proven. Indeed, Chapter 3 of the thesis can be viewed as providing a formal framework that instantiates the concepts described by Maurer and Renner [MR11, Mau11]. This section contains a high-level overview of the concepts and this framework.

1.1.1 The Construction Paradigm

A central and well-known paradigm in constructive disciplines is to construct a complex system from simpler component systems or modules, which each may consist of yet simpler component systems, and so on. The TCP/IP protocol stack, for instance, can be interpreted in this respect. The components are communication resources, starting from different types of physical connections, and resulting in a world-wide communication network. The media access layer (with protocols like Ethernet, Token Ring, or PPP) constructs from a physical channel a point-to-point communication channel between two nodes. The IP layer constructs from an incomplete network of point-to-point channels a communication network where any pair of parties can communicate via sending messages unreliably. The TCP layer then constructs from this resource a *reliable* communication network for bit streams. One main advan-

tage of this approach is modularity, which ensures that each sub-protocol can be designed and analyzed independently from the other sub-protocols.

Constructive cryptography as proposed by Maurer and Renner [MR11, Mau11] applies the described paradigm in cryptography. The components of interest are the *resources* that are available to the parties that participate in the setting where the protocol is used. The resources we consider in this work include different types of communication channels and shared secret keys, but more generally any kind of "infrastructure" that can be used by protocols, like memory or computation, is considered a resource.

The security of a cryptographic protocol is then defined by explicitly modeling both the application in which the protocol is supposed to be used and the desired guarantees the protocol is supposed to achieve in terms of resources that are available to the parties. The goal of a cryptographic protocol π is described as *constructing* the desired resource S from the assumed resource R, denoted as $R \xrightarrow{\pi} S$. Any two such construction steps can be composed, that is, if we consider another protocol ψ that assumes the resource S and constructs a resource T, the composition theorem states that

$$R \xrightarrow{\pi} S \quad \wedge \quad S \xrightarrow{\psi} T \quad \Longrightarrow \quad R \xrightarrow{\psi \circ \pi} T,$$

where $\psi \circ \pi$ denotes the composed protocol.

The composability of constructions allows for a modular design of protocols as a sequence of construction steps. A security proof guarantees the soundness of one such step, and each proof is independent of the remaining steps. The composition theorem then guarantees that several such steps can be composed. A consequence of this approach is that each cryptographic scheme (or security mechanism) is independently analyzed as a construction, which not only simplifies and modularizes the security *proof*, but moreover advocates a modular protocol *design* in which sub-protocols can easily be replaced or modified, without the need to change or re-prove the remaining protocol steps.

1.1.2 Abstract Cryptography: The Axiomatic Formalization

The foundational idea of abstract cryptography as introduced by Maurer and Renner [MR11] is to phrase all definitions, statements, and proofs in the most abstract, and hence formally simplest, formulation. This approach, which originates in mathematical disciplines such as algebra, is *axiomatic* in that one starts with defining abstract structures and continues by describing more concrete structures that satisfy the axioms of the abstract structures. The approach can be exemplified using group theory.

A *group* is an abstract mathematical structure which is described by a set of objects (the group elements), operations on the group elements, and axioms on the operations. The group operation is a binary operation, inversion can be viewed as a unary operation, and the identity element can be viewed as a nullary one. Several statements can be phrased and proven at this abstract level. For instance, one can prove that a group of prime order is necessarily cyclic. This statement, if phrased at this level, transfers to all concrete structures that satisfy the group axioms.

The next lower level is concerned with concrete groups, such as the set $\mathbb{Z}_n$ of all integers modulo some number $n \in \mathbb{N}$. This level requires one to work with a "more detailed" type of object—each group element is an equivalence class of integers. All statements from group theory are inherited, and furthermore one can make additional statements. For instance, one can prove that the group $\mathbb{Z}_4$ is cyclic (and hence different from the direct product $\mathbb{Z}_2 \times \mathbb{Z}_2$), a statement for which it is necessary to investigate the concrete operation on the concrete group elements.

An even lower level may deal with the implementation of a particular group and its operations on a computer. At this level, one describes the elements of a specific group (say, $\mathbb{Z}_n$) in some representation in a specific computational model, usually as bit strings. Algorithms that compute the group operations operate on this representation. This level also fixes a concrete computational model and a notion of run-time. If the implementation is (shown to be) correct, all statements proven at the upper levels still apply. Moreover, at this level one can formulate and prove statements like that the group operations can be computed efficiently, and prove or conjecture that certain problems in the group are hard to solve in the specific computational model.

Abstract cryptography introduces a similar abstraction hierarchy in cryptography. At the highest level of abstraction, the considered objects are *abstract systems* [MR11], and formalize the concept of objects that can be composed by connecting interfaces. This comprises objects like algorithms or automata; the interfaces correspond to the input and output of the algorithm, and the messages obtained or provided by an automaton. Mathematically, an abstract system is an object that has interfaces which are labeled and via which it can be connected to other systems. The resulting object is again an abstract system; the labels of the interfaces allow to describe in which way two objects are composed. In this sense, abstract systems form an algebra where the systems are the objects and connecting systems via their interfaces is the operation; the particular interface sets and connecting operations defined in an algebra determine which topologies can be expressed. In most system algebras, the order in which systems are composed is irrelevant; such algebras are called *composition-order invariant*. The concept of a system alge-

bra is formalized in Section 3.2.

The next lower level is concerned with objects that capture the behavior of systems and formalize the interaction via their interfaces. The most important type in computer science is that of a *discrete system* where different systems communicate by passing messages. Each system operates in a sequence of activations, where in each activation the system first receives messages via its interfaces and subsequently provides as output messages at its interfaces. Each message is then delivered to the system that is connected to the interface where the message was output. The concept of discrete systems exactly describes the input-output behavior. Other instantiations at this level may, for instance, formalize systems where the communication is analogue.

Computation, that is, the concept of *how* the outputs of a system are computed from the inputs, is formalized at an even lower abstraction level. At this level, objects like Turing machines, random access machines, and algorithms appear, and notions like run-time and efficiency are defined. Of course, for each concrete model considered at this level, one has to show how the objects can be seen as discrete systems, and that the model satisfies the axioms of the upper levels.

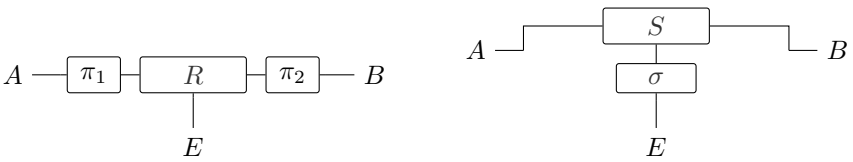
Such an abstract treatment has two major advantages. The first one is *simplicity*: Statements that are phrased and proven at the abstract level do not need to involve technical details such as *how*, for instance, the composition of algorithms is defined; proofs are obtained by syntactic operations on the expressions. The second advantage is *generality*: An abstract proof extends to a given instantiation of the system algebra under the sole assumption that a composition operation is defined and satisfies the axioms. This allows to prove, e.g., the composition theorem of constructive cryptography once and for all; it automatically transfers to system algebras based on conditional probability distributions (for information-theoretic statements) or any concrete type of algorithms (for complexity-theoretic statements) for which the axioms are fulfilled.

1.1.3 Constructions for Secure Communication

The objects that appear in a construction statement, that is, the cryptographic protocol and resources assumed by this protocol, are described in terms of abstract systems in a so-called *cryptographic algebra* [MR11]. This algebra consists of two sets: a set of *resources* which model the realistic resources and which are systems that provide one interface for each party in a given scenario, and a set of *converters* which model the protocol engines used by the parties; a protocol is a tuple containing one converter for each party. A converter is a system that has two interfaces: the *inside* interface that connects

to the party's interface of the resource, and the *outside* interface that is used by the party instead of the original interface of the resource.[1] Roughly, a converter can be seen as translating the actions that the party wants to perform on the constructed resource (e.g., send a message securely) into actions on the assumed resource (e.g., encrypt the message and send it insecurely), and the composition operation defines how a converter is applied to a resource.

The notion of construction. The basic scenario in the context of secure communications is described by two (honest) entities A and B that want to communicate securely in a potentially hostile environment. Consequently, we consider resources with three interfaces: two interfaces labeled A and B for the honest entities, and a third one that is labeled E and captures potential adversarial access. A protocol $\boldsymbol{\pi}$ is a pair of converters, one converter π_1 for A and one converter π_2 for B. We use the term $\boldsymbol{\pi}R$ to describe the scenario in Figure 1.1a, where A uses π_1 and B uses π_2.



(a) A resource R with converters π_1 and π_2 connected at interfaces A and B, respectively. This setup is denoted as $\boldsymbol{\pi}R = {\pi_1}^A{\pi_2}^B R$.

(b) A resource S with a converter σ attached at the attacker's interface E. This setup is denoted as $\sigma^E S$.

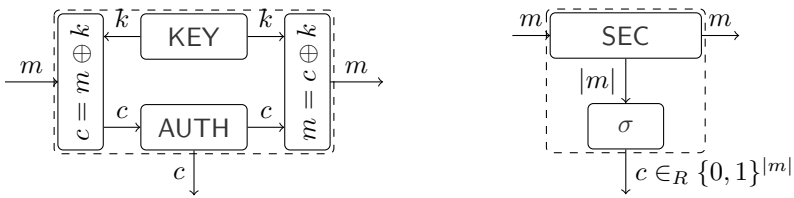
Figure 1.1: The two settings involved in the definition of construction. The algebraic expressions describing such system structures are explained in more detail in Section 3.2.2.

Following the constructive paradigm, the goal of a protocol $\boldsymbol{\pi} = (\pi_1, \pi_2)$ is to construct a certain desired resource S from an assumed resource R. The converters π_1 and π_2 can be seen as translating the actions that A and B want to perform on the desired resource S into actions on the assumed resource R. The protocol is secure if all actions that a potential attacker can perform on the resource $\boldsymbol{\pi}R$ via its interface can be transformed into actions on the resource S; if all attacks on $\boldsymbol{\pi}R$ can be translated into corresponding attacks in

[1] This corresponds to a star-shaped topology: the resource is the central object, which is accessed from the parties. Hence, the interface of the converter that connects to the resource is at the "inside," the interface provided to the party is at the "outside."

the setting with S, then the security guarantees achieved by $\boldsymbol{\pi} R$ are at least as good as the guarantees provided by S. As S specifies the guarantees desired by A and B, the protocol $\boldsymbol{\pi}$ achieves its goal and is said to *construct* the resource S from R. The translation of the attacker's actions is formalized using a so-called simulator σ, which is also a converter and is attached to the E-interface of S. This setup with the constructed resource S and the simulator σ is depicted in Figure 1.1b. The validity of each construction step is hence "locally" defined as an ideal-world/real-world experiment similar to previous definitions such as [Can01, PW01].

Example: the one-time pad. The purpose of the simulator is best understood by considering a simple example. We reproduce in Figure 1.2 the application of the one-time pad in a simplified setting as described by Maurer [Mau11]. A more detailed discussion is provided in Section 4.4.1. Figure 1.2a depicts the application of the one-time pad: We assume as resources a key KEY, which provides to both A and B a uniformly random bit string k, and an authenticated communication channel AUTH, which allows A to transmit a ciphertext to B such that the transmitted value may be leaked to E but cannot be modified. The sender A encrypts a message m by computing a bit-wise XOR with the key k. This results in a ciphertext c that is transmitted over the authenticated channel to B. The receiver B decrypts the ciphertext c analogously to the encryption by computing a bit-wise XOR with k. It is easy to see that this results in the original message m. The attacker at the E-interface obtains the ciphertext c, but cannot influence the protocol further.



(a) The one-time pad applied to an authenticated channel.

(b) The constructed resource with a simulator.

Figure 1.2: The one-time pad as an exemplary application.

Figure 1.2b depicts the "constructed" resource SEC: The secure channel transmits the given message m from A to B, leaking only the length $|m|$ of the message at the E-interface. Clearly, the E-interface of SEC differs from the one shown in Figure 1.2a: while at the authenticated channel the attacker observes a bit string c of length $|m|$, at the secure channel he observes the

number $|m|$ that specifies the message length. This difference can be remedied by using a simulator σ that, given the message length $|m|$, outputs a uniformly random bit string $c \in \{0,1\}^{|m|}$ of the same length as the message. The security proof now proceeds by observing that the two "dashed" boxes in Figure 1.2 describe the same discrete system. The intuition underlying the simulation paradigm is that access to the channel SEC is "as useful to the attacker as" being in the situation of Figure 1.2a, since an attacker can simply sample the ciphertext by himself, obtaining a ciphertext with the same distribution, and then launch the same attack as on the real scheme.

More general settings. For several statements we show in this thesis, the above described scenario with only two honest parties A and B is not sufficient. This includes notions of anonymity where additional "potential participants" need to be present to even formulate the guarantee, and the setting for unilaterally authenticated protocols in which one authenticated entity communicates with one (or more) unauthenticated entities from a population of potential communication partners. Such scenarios can be captured by formulating resources that provide more than two "honest" interfaces and protocols that consist of multiple converters.

Parametrized constructions. Security statements about cryptographic protocols are often parametrized with values that may depend on the scheme, such as the key length of an encryption scheme, or on the environment in which the protocol is used, such as the number of messages that are encrypted or the protocol participants that are corrupted. The parameters determine the exact bound that can be proven for the cryptographic scheme. In a constructive perspective, this is formalized by making the parameters properties of the resources, and proving that a protocol constructs a *parametrized family of resources* from another parametrized family of resources.

The advantage of reflecting the parameters in the resources, instead of in the distinguisher or the adversary as done in most models, is that the construction statements can still be formalized only in terms of the "objects" they consider (i.e., the resources, the protocol, and the simulator), and need not explicitly refer to a restricted class of distinguishers or adversaries. This implies that the standard composition theorem extends directly to the case of parametrized statements, and the "syntactic" composition of sub-protocols extends to this case.

1.1.4 Discrete Systems and Reductions

Abstract systems are abstract mathematical objects and capture the topology in which multiple systems are connected. To describe (and prove the security of) cryptographic protocols, we need to specify the *behavior* of these objects. In the usual setting considered in cryptography, the systems are *discrete* in the sense that their activity can be viewed as occurring in distinct phases, where each such phase consists of taking as input one or more messages—elements of some set—and potentially producing as output one or more messages. In a fully asynchronous setting, the systems are *reactive* in that they become active only upon receiving some input. We allow systems to be *probabilistic*, and they may be stateful in the sense that if their behavior spans multiple phases, then outputs produced in some phase may also depend on inputs obtained in previous phases.

For the case where the order in which the inputs are given to a system is predetermined, random systems as introduced by Maurer [Mau02] exactly formalize discrete behavior. The scenario of secure communication, however, involves several independent parties that communicate, and the order in which messages are received is not necessarily determined in the beginning. The most general type of discrete system describing arbitrary types of behavior, however, does not allow for a modular description of complex scenarios because the order in which messages are delivered may have a crucial influence on the behavior [BA83, Bro83]. As most protocols and resources we consider do not depend on the order of the inputs, but only on their value, we restrict our attention to a more specific type of system, which can be seen as a probabilistic version of the type described by Kahn [Kah74]. In Section 3.4, we extend the random systems concept to this more general case and show that this type of system indeed fulfills the axioms of the system algebra.

At a lower abstraction level, one can define an explicit model of computation and prove that such a model instantiates the discrete systems described here. On that layer, one can then conjecture (or prove) the hardness of computational problems. An obvious question arising here is: without specifying such a computational model, how can we (claim to) make statements about cryptographic schemes that are secure only under computational assumptions? The answer is that our proofs are formalized as *explicit reductions*: the validity of a construction statements corresponds to a distinction problem for two settings, and we explicitly describe for *any* distinguisher how it is translated into a solver for a computational problem, and how the performances of the distinguisher and obtained solver relate.

As the statements we prove extend to *any* computational model that instantiates the discrete systems layer, they prove that in any such model the probability of breaking the scheme is bounded by a related probability of solv-

ing the underlying computational problem. Consequently, if one conjectures (or even proves) such a computational problem to be hard in some specific model, then the proof implies that the scheme will be secure with respect to all attacks that can be implemented in that model with comparable efficiency.

1.1.5 Related Work on Paradigms for Defining Security

The framework we describe differs in various aspects from previous definitions used in the cryptographic literature. The arguably most significant aspect of a security definition is the paradigm employed for defining what a cryptographic scheme is supposed to achieve. Intuitively, the main question that must be answered is "*When should a cryptographic scheme be considered secure? How does one define what security means?*" In this section, we review previous work of security definitions and compare the underlying approaches to the constructive paradigm.

Attack-based definitions. Traditionally, cryptographic security is defined and analyzed in an attack-based view, where security means that, for a certain notion of attack, an attacker with well-defined resources cannot launch a successful attack against the cryptographic scheme. An explicit formalization of this approach has been provided by Yao [Yao82] and by Goldwasser and Micali [GM84]; the technique reached its current state in to work of Bellare and Rogaway [BR06]. The basic idea of these definitions is to describe a game as the interaction between a hypothetical *challenger* and a hypothetical *adversary*. The adversary has access to a certain set of "oracles" provided by the challenger, which model the supposed use of the cryptographic scheme in applications. For instance, to prove an encryption scheme secure against chosen-plaintext attacks, the adversary would be given an oracle to encrypt plaintexts of his choice. The overall goal of the adversary is defined by a certain *winning condition*; depending on the purpose of the considered scheme, this condition may correspond to, for instance, extracting useful information from a ciphertext or forging a digital signature. If there is no (efficient) adversary for which the probability of satisfying the winning condition is noticeably higher than for a trivial strategy such as guessing the correct answer, then the scheme is said to fulfill the property defined by the game.

An advantage of these security definitions is that they tend to be technically simple and are common in many areas of cryptography. It is, however, usually not clear whether a system composed of several cryptographic schemes, each with an individual attack-based security proof, is secure for an overall attack-based security definition. In other words, security often does not compose in a meaningful way. One instructive example is given by

Bellare and Namprempre [BN08] and Krawczyk [Kra01], where it is shown that a natural composition of a secure encryption scheme and a secure authentication scheme fails in providing both confidentiality and authenticity. Another example is the early security definition for quantum key distribution. As discussed by Renner [Ren05], this definition did not guarantee that the obtained key can be used securely, even in the simple case of one-time pad encryption. Indeed, in an attack-based view, proving the security of a composite scheme that consists of several sub-schemes requires one to explicitly show a reduction showing that breaking the security of the composite scheme requires breaking the security of one of the sub-schemes. In constructive cryptography and models based on the idea of universal composability [Can01], an explicit such proof is not necessary because the security of the composite scheme follows generically from the composition theorem.

Many different properties have been proposed in the literature, which raises the question of which security notion for a given scheme is suitable or necessary for a certain higher-level protocol (using that scheme) to be secure. The traditional answer to this question is that for each protocol one (actually, a cryptography expert) needs to identify the right security notion and provide an explicit reduction proof, showing that breaking a certain security property of the protocol requires breaking the security of the underlying scheme. This is mostly due to the fact that the models fall short of arguing why they are sound abstractions of reality; it is difficult to argue that the oracles provided to the adversary formalize the use of the scheme in a realistic scenario. For instance, even modifications such as padding plaintexts to a certain length before encrypting may render proven schemes insecure, which has been modeled in the game-based setting by introducing additional "padding oracles" [Vau02] after such attacks occurred. A further example is described by Hirt and Zikas [HZ10], who show that the property-based definition of broadcast does not achieve the intuitively expected security guarantees. These examples show that the actual security guarantees that one obtains by applying a provably secure scheme in a specific context are usually not evident. Finally, game-based notions are fragile in the sense that seemingly innocent changes may have a significant impact on the security guarantees. One example is the security notion *indistinguishability from random bits* introduced by Rogaway et al. [RBB03]. A minor change of the way in which the random ciphertext is sampled [Iwa06] makes the guarantee meaningless.

Idealization of schemes. A different type of security definition is based on idealizing the behavior of a given scheme. This type is based on the "real-world/ideal-world" paradigm as introduced by Goldreich et al. [GMW87] and formalized in the subsequent works of Goldwasser and Levin [GL90], Micali

and Rogaway [MR91], and Beaver [Bea91] in the context of multi-party computation (MPC). Those definitions are often referred to as *simulation-based*, since they are based on the simulation paradigm introduced in the seminal work by Goldwasser, Micali, and Rackoff [GMR85] on interactive proofs. A protocol is deemed secure if, for each adversary that attacks the protocol, there exists a simulator (also called "ideal adversary") that implements the same attack in the hypothetical execution of an idealized process that is secure by definition, in the sense that no (efficient) algorithm can distinguish between the transcripts (i.e., all inputs, outputs, and messages appearing in the protocol run) of the two executions. The underlying intuition is that, since every attack can be translated from the real protocol execution to the ideal process, the protocol is at least as secure as the ideal process. Several specific simulation-based definitions appear in the literature for tasks such as key establishment [BCK98, Sho99]. These definitions often do not come with explicit composition theorems and are usually referred to as "stand-alone" security.

Based on the initial simulation-based definitions, several frameworks that allow for a more general composition of protocols have emerged. The first and still most widely used such framework has been defined by Canetti [Can01], other proposals have been made by Backes, Pfitzmann, and Waidner [PW01, BPW07] and by Küsters and Tüngerthal [Küs06, KT13]. In these frameworks, a cryptographic scheme, which is formalized as a tuple of algorithms, is shown to be as secure as an idealized version of the algorithms, which is often called the *ideal functionality* and models the goals and the tolerated imperfections of a scheme. The ideal functionality is described with an explicit adversarial interface with well-defined capabilities to model the security guarantees. While the mentioned frameworks differ considerably in many technical details such as the underlying model of computation, they all build on the same idea. Essentially, the distinguishing algorithm also present in the prior simulation-based definitions becomes interactive in the sense that it does not only compare the transcripts of the completed real and ideal executions. Instead, it also influences the ongoing execution by providing local input to or receiving local output from the protocol machines as well as seeing and potentially changing the messages sent during the execution.

The above described approach leads to capturing the functionality of cryptographic schemes in ideal functionalities (sometimes also called ideal interfaces [BFK+13a]), such as the public-key encryption functionality $\mathcal{F}_{\text{PKE}}$ described by Canetti [CKN03] or the functionalities for symmetric encryption described by Küsters and Tüngerthal [KT09]. The interfaces of such a functionality closely resemble the interfaces of the algorithms (although, e.g., the private keys are never output). In such a treatment, elements that are es-

sential for using the scheme, such as the ciphertext or the public key, will still appear in the functionality, but they are idealized in that, for example, the ciphertext is independent of the corresponding plaintext; the idealized scheme is unbreakable by definition. The most significant difference between this type of security definition and constructions is that an idealized algorithm still provides an interface that resembles an implementation, whereas a constructive statement models the application of a scheme in its supposed environment. This effects that, first, the exact assumptions required for securely using a cryptographic scheme are still not explicit in the security statement, and that, second, a higher-level scheme will still depend on the actual *type* of the lower-level schemes, not only their provided guarantees. While from a constructive perspective it is irrelevant whether an authenticated channel assumed by some scheme is constructed via a MAC or a signature scheme, the same statement does not hold if one formulates it with respect to the idealized algorithms. That implies that a higher-level scheme cannot be designed independently of the lower-level schemes.

The formal frameworks underlying the described works can be used in (or easily adapted to) security statements of a different type; one can (almost) make constructive security statements within the formal frameworks of [PW01, Can01]. The statements shown in [PW01, CK02b] involve functionalities which are similar in spirit to the resources we describe here, and the security statements can (almost[2]) be viewed as constructions.

1.1.6 Related Work on Models of Discrete Systems

Most schemes considered in cryptography are *discrete systems* in the sense that their behavior is described by taking as input messages, i.e., elements of some well-defined message space, and as a reaction producing as output also messages which are then given as input to other systems. Various models of discrete systems appear in the literature on distributed systems and cryptography. Most of these models are based on concepts from related areas of computer science like complexity theory or distributed systems. As a result, the models often focus on unsuitable aspects, and additional artifacts are introduced by artificial design choices which are made when adapting the models to the cryptographic setting. By contrast, our goal is to capture the

[2]This would mean that a protocol constructs an ideal functionality from "assumed ideal functionalities," often referred to as hybrid functionalities. As protocols can instantiate hybrid functionalities during the execution and even with adaptively chosen code, one cannot specify a protocol independently of its hybrids (as it would be necessary for a constructive statement). In general, it is even impossible to decide whether a protocol is an "$\mathcal{F}$-hybrid protocol" for some $\mathcal{F}$. If one changes the handling of hybrid functionalities, one can interpret the statements from a constructive perspective.

(conceptually minimal) mathematical type of discrete systems.

There is an important conceptual difference between the *type* of an object and the *specification language* one uses to describe the object. As an example, consider the mathematical type of a function $f : \mathcal{X} \to \mathcal{Y}$ for two arbitrary but fixed discrete sets $\mathcal{X}$ and $\mathcal{Y}$. Generally, there may be different *a priori* equally valid ways to specify what f does: if $\mathcal{X}$ is finite, one could write down the function table; if $\mathcal{X}$ and $\mathcal{Y}$ carry some algebraic structure, there might be a closed algebraic expression for f; if f is computable, one could write down an algorithm computing f, and so forth. The only requirement for a specification language is that each specification *uniquely* describes the object that is to be considered. In contrast, the type (here: the function) is universal and independent of the description. This separation between type and specification language is often unclear in existing models.

Describing individual systems. As most practical cryptographic schemes and protocols rely on complexity-theoretic assumptions, many models that appear in the cryptographic literature are based on the Turing machine model inherited from complexity theory. For instance, many game-based models define a random experiment based on a Turing machine that has access to a set of oracles (a standard technique in complexity theory): the adversary is a Turing machine, and its access to the scheme is formalized via oracles. Several general frameworks such as Canetti's UC model [Can01] and subsequent works such as [Küs06, HS11] are based on a setting with multiple communicating interactive Turing machines (ITMs), which are basically Turing machines that can additionally write to tapes of other Turing machines.

While the objects are defined to be ITMs, they are usually described using some type of pseudo code. Without fixing a rule of how the pseudo code is to be translated into a Turing machine, the pseudo code does, however, *not* even exactly specify the computation that an ITM performs. Moreover, the exact (mathematical) definition and properties of ITMs are hardly used, even in the fundamental theorems concerning the model. The intuitive understanding, made explicit by Hofheinz and Shoup [HS11], seems to be that "[...] none of these details matter." But then, this can also be seen as evidence that ITMs are not the suitable abstraction.

I/O automata discussed in the literature [SL94, PW01, CCK+06, BPW07] are more abstract than a description in terms of Turing machines. In particular, an automaton is an object with state and is described by the state transition that it makes upon receiving a certain message. Hence, while the modeling of state is still explicit, the exact computation of the state transition does not need to be specified. Often, an algorithmic language is used to specify the transition functions.

The work of Kahn [Kah74] models processes in a network as functions from input histories to output histories; several processes are connected by identifying the output values in the history of one system with the input values of another system. For this approach to work, one has to require a certain type of monotonicity from the functions, which restricts their expressiveness in the sense that certain types of behavior cannot be captured. More concretely, systems may not behave differently based on the order in which messages on "different input channels" arrive. Kahn also describes a specification language which is carefully designed to allow only for describing objects of the considered type. The type of discrete system we consider in this work is similar to the one described by Kahn, but additionally captures probabilistic behavior.

The recent framework of Micciancio and Tessaro [MT13] can be seen as a generalization of Kahn networks. Systems do not need to exactly fit the type described by Kahn; intuitively an output of a system can be replaced by an "overdefined" value if two differing orders of inputs would lead to different values for the output. The objects they consider are functions which are monotone as in Kahn's model, but monotonicity is more general for their type. Still, the model is geared towards being a proof technique, the *type* considered (an extended type of monotone functions) is not the actual type of a protocol. Furthermore, they do not provide an interpretation for computational statements.

A different approach underlies the code-based game-playing technique of Bellare and Rogaway [BR06], where manipulations of code (i.e., the specification language) are a core part of the framework. This allows for a rigorous specification and proof of schemes, but the type of behavior is not described explicitly. Furthermore, it is not clear how their model (which is specified for game-based definitions) extends to topologies which are more general than the interaction of a single adversary with a single challenger.

The goal of the random systems framework [Mau02, MPR07] is to exactly capture the type of "probabilistic and interactive" objects like (uniform) random functions, (uniform) random permutations, block ciphers, MAC functions, and so forth. These objects, however, have the property that the interaction with their environment is specified by a sequence of queries with a certain pre-determined order—they can be seen as single-interface systems. Random systems do not immediately give rise to a general instantiation of an abstract system algebra, as in a general setting it is not guaranteed that the inputs are given in the assumed order. One can still specify a system algebra by always providing the interface at which an input (or output) is given explicitly in the provided value, thereby also restricting the system to provide at most one output in each activation, which is similar to the type of communication

described in the UC framework [Can01].

Interactions of multiple systems and scheduling. In a general setting with asynchronous systems, events may occur in different orders depending on how long it takes for messages to be delivered to the receiver. To make the ordering of events in a protocol execution well-defined, the models of [Can01, Küs06, HS11] restrict the type of system to generate at most one message in each activation and remain inactive until it receives the next message. This allows for easily determining the next ITM to be activated, since at every point in time only one ITM has input (this is always the receiver of the message, hence all activations build a chain). Additionally, there is a "master-scheduler" that is activated if no output is generated. While the constraint on the systems is necessary for the particular machine model and scheduling scheme, it restricts the type of systems that can be described and necessitates further artifacts such as, e.g., "delayed outputs" that are scheduled by a dedicated adversarial entity (see Canetti's UC framework [Can13]).

Models based on automata often allow for producing a variable number of output messages in each step, which requires them to define a scheme for the scheduling of the messages; the employed schemes differ between the models. The models related to reactive simulatability (RSIM) [PW01, BPW07] determine a dedicated scheduler, a system that decides in which order the other systems are activated. Often, this role is taken by the adversary to make a "worst-case assumption." Still, such an explicit scheduler that globally controls the activations of the automata is an artificial concept that does not have a counterpart in the real world. Other frameworks define more complicated schemes: in the framework of task-structured probabilistic I/O automata [CCLP07], several different "levels" of scheduling are specified: The "large-scale" schedule of *tasks*—a collection of several "associated" actions of an automaton—is chosen non-deterministically in advance, while the schedule within these tasks is determined by an explicit local scheduler.

Run-time, efficiency, and asymptotics. Most frameworks come with a fixed notion of run-time and efficiency [Can01, Küs06, HS11], which is usually some variation of polynomial time. As the standard complexity-theoretic definition (the run-time of an algorithm must be polynomial in the size of the input) is not applicable to the more general scenario of interactive Turing machines, the frameworks generalize that notion. Straightforward approaches such as requiring the run-time to be polynomial in *all* inputs do not work, since two systems could "ping-pong" messages and run indefinitely. Multiple types of definitions have been discussed, most of them introducing further

artifacts to the formal model. The notion used in the widely used UC framework [Can01, Can13] is based on the idea that inputs "contain" run-time and hence the length of inputs to other machines is required to shrink throughout the execution. This implies that one needs to (formally) modify protocols and append artificial suffixes for most inputs to transmit "run-time" to the receiver of that input. The definition of Küsters [Küs06] introduces an asymmetry to the type of system such that only some subset of the tapes is "enriching" and the run-time of the system must be polynomial in the length of the contents of those tapes; the structure of a composite system must not contain a cycle in terms of enriching tapes. A more indirect approach has been given by Hofheinz, Müller-Quade, and Unruh [HMU05] and is adopted by Küsters and Tüngerthal [KT13], who define that a protocol is efficient if it makes polynomially many steps in any (strictly) poly-time environment.

The complexity-theoretic approach to defining run-time as the number of steps an algorithm requires on a Turing machine has several downsides. Measuring run-time in terms of Turing machine steps only gives somewhat limited information about the actual run-time in a more realistic model (such as a random access machine), because various steps are significantly less efficient on a Turing machine. If one is interested only in statements with respect to polynomial run-time, this artifact is not important because it is known that most computational models are equivalent "up to a polynomial gap." Yet, for obtaining practically meaningful cryptographic statements, a "concrete" type of run-time, measuring e.g. the effort that it takes to break a cryptographic scheme for some *fixed* key length, is required, because concrete parameters must be chosen before a scheme is deployed in practice. While some schemes such as the one of Canetti [Can13], despite their artificial definition, allow to at least obtain *some* insight about concrete run-time, other notions such as the one from Hofheinz et al. [HMU05] do not even have an interpretation for concrete values but are inherently bound to be asymptotic. The asymptotic parameter is usually an input to the algorithms, either on a special "security parameter" tape or as part of the usual input. The security parameter is often specified in unary to make use of standard complexity-theoretic definitions of run-time, which is measured in terms of the input length. If a model (such as ours) is not intrinsically asymptotic but one is interested in making asymptotic statements, one can still consider families of concrete objects for each $k \in \mathbb{N}$. This corresponds to "non-uniform" algorithms; uniformity can be made explicit as a requirement at the level at which computation is defined.

1.2 Constructing Resources for Secure Communication

The second contribution of the thesis is the application of the formal framework described in Section 1.1 to cryptographic protocols in the area of secure communication. The first application we consider is the protection of message transmissions in a setting where a shared secret key is available to the honest parties. The most important types of schemes in this scenario are symmetric encryption schemes and message authentication codes, and there are two ways of composing the schemes: either one first encrypts and then applies the authentication code, or one first applies the authentication code and then encrypts the authenticated message. The relevant resources and the constructions that are achieved in those cases are sketched in Section 1.2.1. The second application is related to anonymity in the context of secure communication. Suppose one is given a public-key encryption scheme, and the sender sends an encrypted message anonymously to a receiver. Will this scheme preserve the anonymity, or will the ciphertext leak information about which key was used to encrypt? The resources required for formalizing this problem and the proposed solution are introduced in Section 1.2.2. The third application is unilaterally authenticated key establishment, that is, key establishment in a setting where only one out of the two participants has a certified public key. We describe the key constructed by such a protocol as a resource and show how to construct it from realistic resources using a simple protocol that is modular, efficient, and secure under mild assumptions, in Section 1.2.3. The material discussed in this section is described in detail in Chapter 4, the major part of it appeared in similar form in earlier publications [MT10, MRT12, KMO+13, MTC13].

1.2.1 Secure Communication Between Two Parties

The basic scenario for secure communication consists of a sender A, a receiver B, and an attacker E. The goal of cryptographic protocols in this setting is to achieve secure communication, that is, communication where the attacker cannot eavesdrop or disturb the communication, even if only non-confidential communication channels are available to begin with.

Resources for Secure Communication between Two Parties

The most important types of resources in the described setting are different types of communication channels, which model both the existing communication resources (e.g., insecure communication via networks such as the In-

ternet) and the desired secure end-to-end communication that cryptographic protocols are supposed to achieve, and that is useful for higher-level protocols such as web browsing, mail transmission, or remote file access. Another type of resource captures the notion of a shared secret key, or other types of shared secret randomness.

Communication channels. A communication channel from an honest sender A to an honest receiver B can be described as a system with three interfaces labeled A, B, and E (the attacker), respectively, where the security properties of the channel are characterized by the capabilities provided at the E-interface. The basic types of channels are (informally) described in Table 1.1, using the notation of Maurer and Schmid [MS96], and are specified formally in Section 4.1.

$- \rightarrow$	An *insecure channel* leaks the transmitted message to the attacker. It also allows the attacker to determine the message output to the receiver. This channel models insecure networks such as the Internet.
$\bullet\!- \rightarrow$	An *authenticated channel* leaks the transmitted message to the attacker. The attacker may forward the same message, or interrupt the channel.
$- \rightarrow\!\bullet$	A *confidential channel* does *not* leak the transmitted messages to the attacker. The attacker can influence the message output to the receiver.
$\bullet\!- \rightarrow\!\bullet$	A *secure channel* does *not* leak the transmitted message to the attacker. The attacker may forward the same message, or interrupt the channel.

Table 1.1: Basic communication channels in the setting with a sender A, a receiver B, and an attacker E.

The intuitive interpretation of the symbol "$\bullet$" is that the capabilities at the marked (sender's or receiver's) side of the channel are provided exclusively to that party. Consequently, if one side is not marked, the attacker might also be able to send or receive messages. We will consider more types of channels, such as channels that allow to transmit only a single message and channels that allow for a larger (or even unbounded) number of messages. A *single-use* channel allows for exactly one message to be input by the sender, and one message to be output at the B-interface, and is indicated by a single arrow tip "$\rightarrow$." A *multiple-use* channel allows for several inputs and outputs; multiple-use channels are indicated by a double arrow tip "$\twoheadrightarrow$."

Keys. Keys that are shared between (honest) parties are also considered a resource. A *shared secret key* is denoted by $\bullet\!= =\!\bullet$ and provides the same random value at the A- and B-interfaces. Depending on the exact type considered, the resource may allow the attacker to prevent the key from being output to the parties, but not obtain the value of the key. This resource models the key that is required by (symmetric) schemes; it could be generated in a key-establishment protocol such as Diffie-Hellman in a setting where both parties are authenticated. Cryptographic keys and other types of shared randomness resources are formally specified in Section 4.2.

Protocols for Secure Communication between Two Parties

Cryptographic schemes such as encryption or MAC schemes give rise to protocols that construct from one type of channel (and possibly a shared secret key) a "more secure" type of channel. In Figure 1.3, the protocol $\mathbf{sc} = (\mathsf{enc}, \mathsf{dec})$ uses as resources a channel $\bullet\!- \rightarrow$ and a key $\bullet\!= =\!\bullet$. The converter enc is attached with its inside interface to the A-interfaces of $\bullet\!- \rightarrow$ and $\bullet\!= =\!\bullet$ (dec is attached to the B-interfaces), and the outside interfaces of enc and dec are the interfaces of the constructed (dashed) system, which is again a channel.

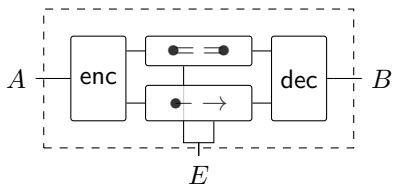


Figure 1.3: Encryption protocol $\mathbf{sc} = (\mathsf{enc}, \mathsf{dec})$ applied to the key $\bullet\!= =\!\bullet$ and the channel $\bullet\!- \rightarrow$.

In Sections 4.3 and 4.4, we show the construction statements achieved by MAC and symmetric encryption schemes. We show that a MAC constructs from an insecure channel (and a shared secret key) an authenticated channel. This construction statement can be written as

$$(\bullet\!= =\!\bullet, - \rightarrow) \quad \xrightarrow{\text{MAC}} \quad \bullet\!- \rightarrow.$$

We show that protocols based on either pseudo-random functions or weakly unforgeable (WUF-CMA—weakly unforgeable under chosen-message attack) MAC schemes are sufficient for this construction. Encryption then corresponds to constructing from an authenticated channel (and another shared

secret key) a secure channel, which is written as

$$(\bullet\!= =\!\bullet, \bullet\!- \!\rightarrow) \xrightarrow{\text{Encrypt}} \bullet\!- \!\rightarrow\!\bullet.$$

We show that the one-time pad (OTP) and cipher-block chaining (CBC) mode encryption both achieve this construction; the one-time pad proof generalizes naturally to stream ciphers and counter-mode encryption. If the ciphertext of a scheme like the one-time pad is transmitted over an insecure channel, the constructed channel will not be secure. We show that, in particular, the one-time pad achieves the construction

$$(\bullet\!= =\!\bullet, - \!\rightarrow) \xrightarrow{\text{OTP}} -\!\oplus\!\rightarrow\!\bullet,$$

where the symbol $-\!\oplus\!\rightarrow\!\bullet$ denotes the XOR-malleable confidential channel, which does not leak the message to the attacker but allows him to modify a transmitted message by applying an "XOR-mask."

Capturing previous definitions. Traditional security definitions in the context of secure communication specify properties of cryptographic schemes in terms of resilience against classes of attacks. For symmetric encryption schemes, these properties are intended to model the protection of the confidentiality or the integrity of the encrypted messages. A vast variety of such definitions has emerged in the literature and, despite the efforts of previous work, the relations and interplay of many of these notions (which are *a priori* not composable) are unexplored. The oracle queries and winning conditions of games encode the use and guarantees only implicitly, and the exact guarantees are often hard to understand.

In constructive cryptography, notions such as confidentiality and integrity appear as attributes of channels, i.e., the communication itself, making the guarantees achieved by cryptographic schemes explicit. The security of communication channels corresponds to restrictions on the capabilities provided at the E-interface, which can be characterized according to two aspects: the amount of information leaked about transmitted messages, and the potential influence on messages delivered to the receiver.

Besides showing that weakly unforgeable MACs construct an authenticated channel from an insecure channel and a shared secret key, we analyze notions for symmetric encryption schemes. In particular, we show that an encryption scheme is IND-CPA (indistinguishability under chosen-plaintext attack) secure if and only if it achieves the construction

$$(\bullet\!= =\!\bullet, \bullet\!- \!\twoheadrightarrow) \xrightarrow{\text{Encrypt}} \bullet\!- \!\twoheadrightarrow\!\bullet,$$

that is, the scheme can be used to protect multiple messages. We also show that an IND-CCA (indistinguishability under chosen-ciphertext attack) secure scheme achieves

$$(\bullet\!\!= \;=\!\!\bullet,\; - \twoheadrightarrow) \quad \xrightarrow{\text{Encrypt}} \quad -\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet,$$

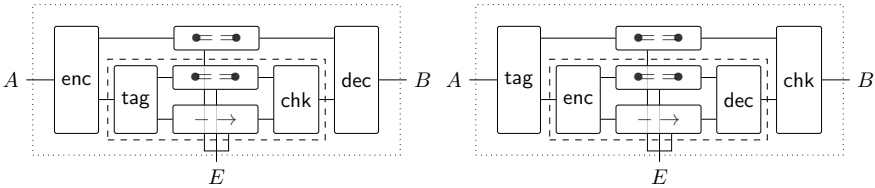
where the channel $-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet$ is confidential but does not prevent, for instance, that messages are replayed or reordered. Also, the channel does not prevent the attacker from sending unrelated messages to B. Finally, we show that a scheme that is both IND-CPA and INT-CTXT (integrity of ciphertexts) secure achieves

$$(\bullet\!\!= \;=\!\!\bullet,\; - \twoheadrightarrow) \quad \xrightarrow{\text{Encrypt}} \quad \bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet,$$

where the channel $\bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet$ protects confidentiality and authenticity but does not prevent replays or reordering messages. In contrast to $-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet$, the channel does not allow the attacker to send unrelated messages to B.

Generic composition of MAC and encryption. The construction steps described above for MAC and encryption correspond to the well-known Encrypt-then-Authenticate (EtA) paradigm. In the dual paradigm, Authenticate-then-Encrypt (AtE), encryption first constructs from an insecure channel (and a shared secret key) a confidential channel, and a MAC constructs from this (and another shared secret key) a secure channel. As pointed out by Bellare and Namprempre [BN00], and Krawczyk [Kra01], there are encryption schemes for which AtE does not achieve the expected guarantees. We show, however, that specific schemes such as the one-time pad or CBC-mode encryption for block ciphers together with a strongly unforgeable (SUF-CMA) MAC are still sufficient to construct a secure channel.

Regarding protocols as transformations of channels is dual to the common interpretation of schemes as transformations of messages. This duality becomes evident in the analysis of the EtA and AtE transformations. EtA is depicted in Figure 1.4a: One first applies the MAC to the insecure channel $- \rightarrow$ and a shared secret key $\bullet\!\!= \;=\!\!\bullet$ (this, as discussed above, can be seen as an authenticated channel), and then uses the encryption scheme to obtain a secure channel $\bullet\!\!- \rightarrow\!\!\bullet$. As in Figure 1.3, the composition of systems grouped by the dashed box behaves as an authenticated channel $\bullet\!\!- \rightarrow$ (with a simulator). An encryption scheme hence guarantees that the (dotted) system constructed from the authenticated channel and a shared key $\bullet\!\!= \;=\!\!\bullet$ is a secure channel. This transformation is usually referred to as Encrypt-then-Authenticate, since the first operation applied to the plaintext input at the outside A-interface is encryption. From our perspective of channel transformations, however, the first mechanism applied to the insecure channel $- \rightarrow$ is the authentication. For the sake of consistency with previous literature, we will maintain the term



(a) Encrypt-then-Authenticate (b) Authenticate-then-Encrypt

Figure 1.4: Generic constructions of secure channels using encryption **sc** = (enc, dec) and authentication **mac** = (tag, chk).

EtA for this transformation, keeping in mind that the permuted appearance of the terms "encryption" and "authentication" is an effect of the paradigm shift underlying constructive cryptography.

The dual paradigm, AtE, uses the encryption to transform the insecure channel $- \rightarrow$ into a confidential channel $- \rightarrow\!\bullet$ indicated by the dashed box in Figure 1.4b. This channel is transformed into a secure channel by a MAC. While EtA is secure under widely used assumptions, the case of AtE is substantially more involved. We show in Section 4.5.2 that the composition of the one-time pad encryption (this extends to stream ciphers) with a strongly unforgeable (SUF-CMA) MAC indeed constructs a secure channel. In contrast to EtA, the second construction step is specific to the malleability of the confidential channel, and in this sense also specific to the encryption scheme.

Related Work

Most research in the context of secure communications follows an attack-based approach as discussed in Section 1.1.5. We describe this related work and sketch the relation to our definitions. For MAC schemes, two slightly different notions of unforgeability (WUF-CMA and SUF-CMA, which we describe in Section 2.4.2) have emerged as the standard notions [BCK96, BKR00]. We show in Sections 4.3.2, 4.5.2, and 5.2 that such schemes can indeed be used to construct an authenticated channel from an insecure one or, alternatively, a secure channel from certain types of confidential ones (always assuming a pre-shared secret key). Bellare et al. [BGM04] have pointed out differences between several variants of the definitions with respect to the number of verification queries that are allowed to the adversary.

The most widely used definitions of confidentiality for symmetric encryption are called IND-CPA and IND-CCA and are derived from the respective public-key notions [GM84, NY90, ZS92], they have been translated to the setting of symmetric encryption schemes by Bellare et al. [BDJR97]. Fur-

ther variants of these notions are introduced and compared by Katz and Yung [KY00a]. We describe the corresponding games in Section 2.4.3 and show in Section 4.4.3 that IND-CPA security is equivalent to constructing a secure channel from an authenticated one, and IND-CCA is sufficient for constructing a non-malleable confidential channel from an insecure one (always assuming a pre-shared secret key).

Several types of integrity guarantees for symmetric encryption schemes have been considered. Notions of non-malleable encryption first appeared for public-key schemes [BDPR98, DDN00], and were later translated to the symmetric case [BN00]. Several related notions such as unforgeability of ciphertexts have been discussed [GDK02, KY00b]. Further widely used notions are INT-CTXT and INT-PTXT (integrity of ciphertext and integrity of plaintext, respectively) introduced and analyzed by Bellare and Namprempre [BN00], their relation is further examined by Paterson et al. [PRS11]. Also, various types of unforgeability notions appear in the literature [GDK02, KY00b, Kra01]. Constructively, integrity guarantees of encryption schemes are formalized by describing the particular type of confidential channel that is constructed by the scheme from an insecure channel and a shared secret key. We show in Section 4.4.3 that a scheme which is IND-CPA and INT-CTXT secure already constructs a secure channel, and as non-malleability (in the CCA case) is equivalent to IND-CCA, a non-malleable scheme constructs a non-malleable confidential channel. Further relations have been discussed by Maurer et al. [MRT12].

Schemes that protect both the confidentiality and the authenticity are often called *authenticated encryption schemes*, and their security is usually defined by a combination of properties for confidentiality and integrity [BN00, RBB03]. The standard combination is IND-CPA and INT-CTXT [BN00, BR00, KY00b]; combinations with weaker types of integrity properties have been discussed in several papers [BN00, CK01, GDK02, KY00b, Nam02]. A single game-based notion for authenticated encryption has been proposed by Shrimpton [Shr04] and Rogaway and Shrimpton [RS06]. Notions of authenticated encryption correspond to the construction of a secure channel from an insecure channel and a secret key. Jager et al. [JKSS12] have introduced *authenticated and confidential channel establishment* in the context of the TLS protocol. The notion formalizes the security achieved by the composition of a key-establishment protocol and authenticated encryption in an attack-based model and was introduced because the “handshake” sub-protocol of TLS cannot be proven in standard models for key establishment. It corresponds to a construction of a secure channel from a resource that formalizes the assumptions of the key-establishment protocol and insecure communication channels.

Simulation-based definitions of secure communication have been given by Pfitzmann and Waidner [PW01] for reactive simulatability, and by Canetti and Krawczyk [CK02b] for universal composability. The ideal functionalities described in those works are similar to the channels we describe here, but the security statements are not phrased as constructions and are formalized in different formal models. Surprisingly, the corresponding security proofs are performed in a single step and do not exploit the composability guaranteed by the respective frameworks. Symmetric encryption has been formalized as an ideal functionality by Küsters and Tüngerthal [KT09]. The description of the ideal functionality turns out to be complex because of difficulties with handling the secret keys within the functionality—these issues disappear if the key is considered as a resource. A "hybrid" approach to the definition of secure channels is pursued in the definition of [CK01]: authenticity is simulation-based, while confidentiality is game-based.

The first formal proof of the one-time pad was given by Shannon [Sha49], who showed that the ciphertext, as a random variable, is independent of the plaintext and therefore does not contain any information about the message. The constructive perspective on the one-time pad has been described by Maurer [Mau11]; we show in Section 4.4.1 how the treatment given there is formalized in terms of discrete systems. Moreover, we also formalize the guarantee that one obtains if the ciphertext is transmitted via an insecure channel. CBC-mode encryption was first formally analyzed by Bellare et al. [BDJR97] in a game-based security model. Their analysis with respect to IND-CPA corresponds to a construction of a secure channel from an authenticated channel and a secret key; we provide an explicit constructive proof in Section 4.4.2. The advantage of the constructive proof is in its modularity: CBC-mode encryption is proven based on a uniform random permutation (instead of a pseudo-random permutation), and the reduction to the security of a pseudo-random permutation follows directly from the composition theorem.

The Encrypt-then-Authenticate transformation of an encryption scheme and a MAC scheme secure under the game-based standard notions is secure as an authenticated encryption, while the corresponding statement does not hold for Authenticate-then-Encrypt [BN00, Kra01]. Results showing that certain combinations of schemes that are still secure exist [Kra01, MT10, PRS11]; as they focus on the schemes used in TLS and hence we discuss them in Section 1.3.3. Ferguson and Schneier [FS03] compare EtA and AtE from a practical perspective. On the one hand, they argue that AtE is favorable for two reasons. First, the MAC is "protected" by the encryption, which makes attacking the authenticity of the combined scheme more difficult. Second, the authentication is applied to the plaintext, while in EtA, only the ciphertext is authenticated. On the other hand, they note that EtA is generically secure

and more resilient to certain Denial-of-Service attacks.

1.2.2 Receiver-Anonymous Communication

A formalization of anonymous communication requires a setting with multiple senders or receivers. The reason is that anonymity generally holds only with respect to some *anonymity set* in the sense that the only information that an attacker obtains is that a message was sent from or to any one of the parties in that set. If this set contains only a single element, anonymity is vacuous. Anonymous communication can protect either the anonymity of the sender or the anonymity of the receiver (or both), and the resources used to formalize the guarantees are therefore modeled with multiple senders, or multiple receivers, or both.

Resources for Anonymous Communication

We focus on the case of receiver-anonymous communication and describe channels with multiple receiver interfaces. Whenever a sender inputs a message for some recipient at its interface, the channel might leak certain information about the message (such as its length or even the complete plaintext) at the attacker's interface; however, the channel will *not* reveal the receiver. We describe two channels that are useful for the analysis of anonymity-preserving public-key encryption in Table 1.2, the specifications are given in Section 4.6.

$- \preceq$	An insecure broadcast channel allows a single sender to broadcast messages to multiple receivers. The messages are leaked to the attacker, who can additionally modify the messages and decide which message is delivered to which receiver. This channel corresponds to, for instance, wireless broadcast transmissions.
$\Rightarrow\!\!\diamond\!\!\Rightarrow\!\!\bullet$	A confidential receiver-anonymous channel allows a single sender to send a message to a chosen receiver. The channel leaks neither the transmitted message nor the identity of the intended receiver to the attacker. The attacker may forward the transmitted messages to the intended receivers, and inject chosen messages to chosen receivers.

Table 1.2: Communication resources with a single sender A, multiple receivers $B_1, B_2, \ldots$, and a single attacker E.

The first channel is an insecure broadcast channel $- \preceq$ that allows the sender to send a single message to all receivers. This channel can be achieved

by either multi-sending via a network of point-to-point channels, or physically as in a wireless communication network. The channel $\Rightarrow\!\!\diamond\!\!\Rightarrow\!\!\bullet$ models confidential and receiver-anonymous communication. In contrast to $- \prec$, the sender chooses for each message a dedicated receiver B_i, and the message will only be output at interface B_i (and not at B_j with $j \neq i$). The attacker learns neither the message nor the receiver.

Protocols for Anonymous Communication

While anonymity and confidentiality appear to be orthogonal properties, making anonymous communication confidential is more involved than one might expect, since the ciphertext might reveal which public key has been used to encrypt. In fact, for the case where encryption schemes are used for end-to-end encryption between senders and receivers, anonymity cannot be *created* by the encryption scheme. Such schemes can only *preserve* anonymity that is guaranteed by the underlying network:[3] If a sender A sends a message to a receiver B over the Internet using B's publicly known IP address, then there is no hope for the encryption scheme to hide the fact that B is the intended receiver of A's message. Encrypting messages potentially makes the problem worse: Even if the transmission of the ciphertext is itself anonymous, the ciphertext might still reveal under which public key it was encrypted.

To address this problem, public-key cryptosystems with enhanced security properties have been proposed. We describe the application of a public-key encryption scheme in the described scenario, and prove in Section 4.6 that the construction

$$(\leftarrow \bullet^n, - \prec) \xrightarrow{\text{PKE}} \Rightarrow\!\!\diamond\!\!\Rightarrow\!\!\bullet$$

is achieved by a public-key encryption scheme that fulfills the properties IND-CCA, key privacy (IK-CCA), and weak robustness (WROB-CCA) defined in the literature, e.g. [BBDP01, ABN10]. The symbol $\leftarrow \bullet^n$ indicates that there is an authenticated channel from each receiver $B_1, \ldots, B_n$ to the sender A which are required to send the public keys of the receivers to the sender, and the channels $- \prec$ and $\Rightarrow\!\!\diamond\!\!\Rightarrow\!\!\bullet$ have been described in Table 1.2.

Related Work

Several types of cryptographic schemes have been investigated with a focus on anonymity. In "key-private" public-key encryption, the ciphertext does

[3]Note that this observation does not hold for active networks or overlay networks that can implement their own multi-hop anonymous routing strategy for which encryption is in fact crucial. Buses [BD03] is a cryptographic design exemplifying this, while TOR [DMS04] is the most widely used anonymity system based on this principle.

not reveal information about the intended receiver [BBDP01, ABN10], in private key establishment [Aba02, CK02a, AF04] two parties can exchange a key without revealing their identities, and "anonymous" signatures protect the signer's identity at least as long as parts of the signed plaintext remain hidden [YWDW06, Fis07].

The first definition of key-private public-key encryption has been given by Bellare et al. [BBDP01]; the goal of the primitive was to attain receiver anonymity. Abdalla et al. [ABN10] noted that also robustness is needed for the PKE scheme to achieve this property, since otherwise an honest receiver is unable to detect whether he is the intended recipient of a given ciphertext and could obtain a bogus decryption. Mohassel [Moh10] analyzed game-based security and anonymity notions for KEM-DEM encryption schemes, showing that, for this particular type of composition, weak robustness together with the key privacy of the KEM (key-encapsulation mechanism) and DEM (data-encapsulation mechanism) components is sufficient to obtain a key-private hybrid public-key encryption scheme. Our result implies that weak robustness is sufficient even for universal composition. However, as shown recently by Farshim et al. [FLPQ13], (even strong) robustness is insufficient in certain contexts, such as Sako's auction protocol. The same concept as weak robustness (i.e., that only the intended recipient must be able to decrypt a ciphertext to a meaningful plaintext) lies at the core of *incomparable* public keys as defined by Waters et al. [WFS03].

Pfitzmann and Waidner [PW85] described several flavors of anonymity in communication across networks, including receiver anonymity. Later, Nagao et al. [NMO08] described similar sender-anonymous channels and showed that such channels can be related by reductions to other types, such as secure channels and direction-hiding channels. Ishai et al. [IKOS06] provided a broader investigation on how to bootstrap cryptographic functionalities using anonymity, based on a particular type of sender-anonymous channel which is similar to the notion of sender anonymity in the framework of Hevia and Micciancio [HM08].

1.2.3 Unilaterally Authenticated Key Establishment

Many practical cryptographic protocols used on the Internet are designed for a client-server setting where only the server has a certified public key. The most prominent example for this use case is access to web servers, but protocols for sending or receiving mail or for accessing database or directory servers often follow the same approach. In these settings, the client and the server generate a cryptographic key which has only *unilateral authentication* (cf. [Sho99, BM03]), i.e., the client is assured to share a key with the assumed

server; the server has no comparable guarantee. The client is later authenticated by sending its credentials, often a username and password, over a connection that is secured with the shared key.

Resources for Unilateral Key Establishment

The goal of a unilateral key-establishment protocol is to construct a *unilateral key*, which is a resource that is denoted as $= \Rightarrow\!\!\bullet$ and guarantees the exclusiveness of the key only at the B-interface. On a high level, this resource provides the guarantee that *either* it outputs the same random key both at the A- and at the B-interface, *or* a key input at the E-interface is output at the B-interface, while the A-interface remains inactive. This is the expected guarantee in a setting where only B can send messages authentically, since the attacker can always block the messages sent by A and engage in the protocol with B, obtaining a key. While party A, once it obtains a key, is guaranteed to share a key with B, the party B has no such guarantee. Hence, without a further authentication step, B cannot distinguish whether it shares a key with A or with the attacker.

We show that the unilateral key is still a useful resource since A can be authenticated later by, e.g., sending a password. The symbol $= \Rightarrow\!\!\bullet$ that we use to denote the resource follows the notation of [MS96]. The marker "$\bullet$" signifies that the capabilities at the B-interface are exclusive to that interface: If a key is output at the A-interface, this key is guaranteed to be shared with the B-interface (and *not* the E-interface). There is no comparable guarantee with respect to the A-interface, and hence there is no "$\bullet$" on the left hand side of the symbol $= \Rightarrow\!\!\bullet$.

The most common authentication assumption made by practical protocols is that a public-key infrastructure (PKI) is available and B has a certified public key. This assumption corresponds to a single-message authenticated channel $\rightarrowtail\!\!\diamond\!\!\bullet$, which allows B to send a single message authentically to all potential communication partners $A_1, \ldots, A_n$, but allows the attacker to prevent the delivery to some or all parties $A_1, \ldots, A_n$. The transmitted message corresponds to B's public key which is certified in the PKI and which is hence transmitted authentically to all parties that can verify the certificate.

A Protocol for Unilateral Key Establishment

We show that the unilateral key is constructed from an authenticated communication channel in one direction (say, from B to A) and an insecure communication channel in the opposite direction by a simple protocol based on a CPA-secure key-encapsulation mechanism (KEM). This construction can be

described as

$$(- \rightarrow, \leftarrow \bullet) \quad \xrightarrow{\text{KEM}} \quad = \bullet.$$

The central idea underlying this protocol is simple: While the traditional approach to use a KEM in this scenario is to let B generate a key pair for the KEM and send the key to A, our protocol works in the opposite direction. A generates a key pair and sends the public key to B, who then encapsulates a key and sends the ciphertext together with the public key via the authenticated channel. This approach allows A to check whether the key has been replaced during the transmission to B. While protocols using the traditional approach require the KEM to be CCA-secure, our protocol is secure if the KEM achieves only CPA security (a much weaker requirement). This is proven explicitly in Section 4.7.

We additionally show that if the parties A and B share a (low-entropy) password, then this password can be used to authenticate the key by a simple protocol in which A sends the password encrypted to B. Denoting the password as a resource $\mathbf{Q}$, this corresponds to the construction

$$(= \bullet, \mathbf{Q}, - \rightarrow) \quad \xrightarrow{\text{Send PWD}} \quad (\bullet = \bullet, \mathbf{Q}) \,.$$

The fact that the password resource $\mathbf{Q}$ appears also as a constructed resource formalizes that the password can be used to authenticate multiple unilateral keys.

Finally, we show how the authenticated channel $\leftarrow \bullet$ assumed by the key-establishment protocol can be constructed if a public-key infrastructure, modeled as a resource $\triangleright\!\!\diamond\!\!\bullet$ that allows B to send a single message authentically to all potential partners, can be constructed in the sense that B obtains one independent such channel to each potential partner. The protocol is based on digital signatures and needs to ensure that each message is only accepted by its intended receiver, and not by other (honest) parties. The achieved construction is

$$(\triangleright\!\!\diamond\!\!\bullet, {}^{n}\!\leftarrow -) \quad \xrightarrow{\text{Sign}} \quad {}^{n}\!\leftarrow \bullet,$$

where the symbols ${}^{n}\!\leftarrow -$ and ${}^{n}\!\leftarrow \bullet$ indicate that there is one channel from B to each A_i for each $i \in [n]$.

Related Work

Various security models for key-establishment protocols have been proposed, most of them in the game-based setting and with a focus on key-establishment protocols with mutual authentication. A partial list includes [BR93, BJM97, BCK98, BM98, CK01, LLM07]. Such security definitions come *a priori* without

composition guarantees, but specific results are known for some of the definitions [CK01, BFWW11]. A simulation-based security definition has first been given by Shoup [Sho99], still without general composition guarantees. A formalization in the UC framework [Can01], which guarantees composability, is given by Canetti and Krawczyk [CK02b].

Only few security definitions apply to the case of unilateral authentication; one early formal treatment of this setting has been given by Halevi and Krawczyk [HK99], with a focus on password-based protocols. Shoup [Sho99] provides the first formal model that fully supports unilateral authentication, and describes several protocols that achieve his security definition; one of those (called A-DHKE) can be viewed as a specialization of our protocol using a Diffie-Hellman based KEM. Goldberg, Stebila, and Ustaoglu [GSU13] extend the so-called eCK-model [LLM07] to support unilateral authentication. The protocol they provide is less efficient (three messages) and based on stronger assumptions (random oracles), but is resilient to attacks not modeled in our definition. Dodis and Fiore [DF13] independently proposed a protocol that is similar to ours. In particular, it is also based on a CPA-KEM and a—there interactive—mechanism for authentication. They also gave a simple game-based security definition for unilateral key exchange. In recent work, Coretti et al. [CMT13b] show that CCA-secure KEMs construct the unilateral key resource in a non-interactive scenario.

Generally, the game-based security models do not come with a proven (or even explicitly stated) composition theorem. The achieved security is, on a high level, comparable: The "key reveal" queries in game-based models correspond in the construction to the fact that the distinguisher obtains (via the interfaces of the honest parties) the keys in *all* sessions. (Intuitively, all sessions can be considered "test sessions," which is stricter than "key reveal" but can be related by a hybrid argument.) The constructive notion of unilateral key establishment does capture static corruption of clients (both passive and active) because, in the unilateral setting, the attacker can initiate sessions with the server by using the capabilities provided at the E-interface of the resources. It does, however, currently not capture adaptive corruptions and also does not model advanced properties such as perfect forward secrecy. Note that since traditional game-based models do not come with a general composition theorem, the guarantees provided to higher-level protocols (and in particular the guarantees with respect to special properties) are, while from an intuitive perspective certainly desirable, currently not formalized.

Most further definitions appear in papers on TLS: Morissey, Smart, and Warinschi [MSW08] extend a standard game-based definition to the unilateral case, but, as in other game-based models [HK99, Sho99, DF13, GSU13], the guarantees this definition provides with respect to composition are un-

clear. The recent analyses of the TLS handshake by Krawczyk, Paterson, and Wee [KPW13] and Kohlar, Schäge, and Schwenk [KSS13] also consider the case of unilateral authentication, but as the unmodified TLS handshake protocol is not secure with respect to "standard" security notions for key establishment, they provide a combined security statement for the complete protocol. Kohlweiss et al. [KMO+14] show that the TLS handshake protocol, without the confirmation messages, constructs (essentially) a unilateral key in the same setting as our protocol. Due to unfortunate design choices in the TLS protocol and its inherently non-modular structure, the proof requires stronger assumptions and a considerably more contrived analysis.

1.3 A Constructive Perspective on the TLS Record Layer

Initially developed as the Secure Socket Layer (SSL) protocol [Hic95] for securing the HTTP communication between web servers and browsers, the SSL/TLS protocol family aims to provide end-to-end security for bidirectional communication over the Internet and is nowadays used in many Internet protocols including, e.g., SMTP or IMAP for transmitting e-mails and LDAP for accessing directories. The typical setting is to only authenticate the server to the client, but client authentication is also possible if the client has a certified public key. The protocol has suffered from several vulnerabilities (e.g., [Rog95, Ble98, Vau02, KPR03, Bar04, RD09, AP12, ABP+13, AP13]); this led to the development of a series of protocol versions and revisions of the implementations, each one fixing flaws discovered in the previous version. The most recent protocol version is known as Transport Layer Security (TLS) version 1.2 [DR08]. A new protocol version 1.3 is currently being developed.

1.3.1 The TLS Protocol

The TLS protocol consists of two parts, the *handshake*—essentially a key-establishment protocol that can be used with either unilateral or mutual authentication—and the *record-layer protocol*—which protects the transmission of application data using the key obtained during the handshake.

The TLS handshake offers several alternative key-establishment methods based on different cryptographic primitives. In each session, a particular method is chosen depending on the implementation, the available public keys, and the configuration. The handshake protocol contains several unfortunate design choices and, because of the non-standard use of schemes

and primitives, papers that analyze (parts of) TLS must generally choose between analyzing a modified version of the protocol, analyzing the original protocol in idealized models (such as the random oracle model), or using tailor-made computational assumptions. A partial list of papers analyzing (parts of) the TLS handshake, each one considering a different subset or variant of the protocol, in different models, and under different assumptions, is [JK02, MSW08, BFCZ12, FHM+12, GIJ+12, JKSS12, BFK+13a, BFS+13, GKS13, KSS13, KPW13, KMO+14].

The main problem in analyzing the TLS protocol is that it was not designed with provable security in mind; it is inherently non-modular and uses cryptographic primitives in non-standard ways. Overall, it is fair to say that the TLS protocol is at the same time not very efficient (for instance, the TLS handshake begins only after the three-message TCP handshake, although a couple of works [Lan10, LMM10, RCC+11, SHI+12] tried to resolve this issue) and has a questionable cryptographic design. The wide use of the protocol is caused by the fact that it is easy to integrate with existing infrastructure; it is based on X.509 certificates and can be integrated by slight extensions of protocols that anyway use a TCP connection for their communication.

1.3.2 Viewing the Record Layer Constructively

The two most widely used types of cipher suites in the TLS record-layer protocol are based on the AtE combination of a symmetric encryption scheme and a MAC. There are two types of encryption schemes: The first one is a stream cipher and is based in RC4. The second one is the CBC mode of a block cipher, the most widely used cipher suites use 3DES or AES. The MAC schemes are generally implemented as using HMAC with a hash function, the specification mentions MD5, SHA-1, and SHA-256.

The initial protocol versions had several vulnerabilities, for instance due to an erroneous chaining in CBC mode [Rog95, Bar04] or error messages of and timing attacks on the MAC verification [Vau02, AP12, AP13]. These vulnerabilities have lead to practical attacks, but have been fixed in the recent versions of TLS. Recently, the doubts concerning the security of the RC4 stream cipher have increased [ABP+13], which makes RC4-based cipher suites a questionable choice. A comprehensive list of attacks can be found in Meyer's thesis [Mey14].

Constructively, the AtE transformation is viewed as first constructing a confidential but malleable channel using the encryption scheme. The malleability of the particular constructed channel depends on the encryption scheme: A stream cipher leads to an XOR-malleability that allows the attacker to modify transmitted plaintexts by applying XOR-masks. The CBC-mode encryption

leads to a type of malleability that resembles the block structure of CBC. The subsequent construction step is achieved by the MAC scheme in combination with a padding scheme and sequence numbers. This step constructs, from the confidential channel and a secret key, a secure channel. This constructive analysis is performed in detail in Chapter 5.

The analysis in our model excludes some, but not all, attacks that have been found against earlier versions of TLS. The vulnerabilities caused by incorrect CBC chaining pointed out by Rogaway [Rog95] and Bard [Bar04], which have later been exploited in the famous BEAST attack [DR11], would have been captured. The so-called CRIME attack, based on the observation that compression ratios for different messages leak information about the plaintext [Kel02] and the original padding oracle attacks [Vau02] are currently not captured because the corresponding parts of the protocol are not analyzed, but would directly show up once compression or error messages are included in the analysis. (This means extending the current analysis, in the same model, to further parts of the protocol.) Attacks which exploit different processing times of algorithms depending on the internal state of the receiver like those shown by AlFardan and Paterson [AP12, AP13] are inherently not covered by the definitions; they would require a modeling of computation time.

1.3.3 Related Work

The two most commonly used cipher suites in the TLS record-layer protocol are based on the AtE combination of a symmetric encryption and a message authentication code. While AtE is not secure for *all* schemes [BN00, Kra01], the combinations of schemes used in TLS is in principle sound. The first result in this direction was shown by Krawczyk [Kra01], who proved that CBC-mode encryption and stream ciphers together with a strongly unforgeable MAC are indeed sufficient. The analysis, however, deviated from the TLS protocol in two aspects: The proof uses a size restriction on the MAC which is not satisfied by the real protocol, and padding in CBC mode is not considered. Our analysis in the conference paper [MT10] and in Chapter 5 handles these issues correctly. The subsequent work of Paterson, Ristenpart, and Shrimpton [PRS11] follows, as [Kra01], a game-based approach, but in contrast to our analysis even takes into account the (optional) variable-length padding of TLS (designed to hide the length of transmitted messages). The bounds proven in all three works are roughly comparable, we provide more details in Chapter 5.

Chapter 2

Preliminaries

This chapter introduces notation and fundamental concepts that are prerequisites for the material in the subsequent chapters. Section 2.1 introduces general notation that is used throughout the thesis. Section 2.2 describes Maurer's abstract reduction theory, which abstracts the reduction concept from complexity theory and which we use to define and prove the security of schemes that are based on computational assumptions. The concept of and foundational statements about random systems, which model probabilistic discrete behavior, are introduced in Section 2.3. Finally, Section 2.4 contains descriptions and definitions of several types of cryptographic schemes along with formalizations of game-based security definitions in terms of random systems.

2.1 Notation

We describe general symbols and writing conventions used throughout the thesis in Section 2.1.1. In Section 2.1.2, we discuss a generalized concept of tuples in which the components are labeled by elements of arbitrary sets, and which underlies several of our definitions.

2.1.1 General Notation

We use the notation $\mathbb{N} = \{1, 2, \ldots\}$ for positive integers and for each $n \in \mathbb{N}$, we write $[n] \coloneqq \{1, \ldots, n\}$. We denote by $\mathbb{R}$ the set of all reals, and use the notation $[r_1, r_2]$ for $r_1, r_2 \in \mathbb{R}$, $r_1 \leq r_2$ to denote the interval between r_1 and r_2, formally $[r_1, r_2] \coloneqq \{r \in \mathbb{R} : r_1 \leq r \leq r_2\}$. For sets $\mathcal{X}$ and $\mathcal{Y}$, we denote the set of all functions $f : \mathcal{X} \to \mathcal{Y}$ as $\mathcal{Y}^{\mathcal{X}}$. For a set $\mathcal{S}$, we denote the *powerset*

of $\mathcal{S}$, i.e. the set of all subsets of $\mathcal{S}$, by $2^{\mathcal{S}}$. If $\mathcal{S}$ is finite, we denote by $|\mathcal{S}|$ the *cardinality* of $\mathcal{S}$, that is, the number of elements in $\mathcal{S}$.

Function names that consist of multiple characters are written in straight serif font such as enc. Special constants such as Boolean values are written in underlined sans-serif font such as $\underline{\mathsf{true}}, \underline{\mathsf{false}}$.

Probability theory. In a given random experiment, the probability that an event E occurs is denoted by $\mathsf{P}(E)$; the random experiment, if not clear from the context, will be specified in the superscript of P. The notation $\mathsf{P}_X(x)$ for a random variable X with values in $\mathcal{X}$ and a value $x \in \mathcal{X}$ then describes the probability $\mathsf{P}(X = x)$. For another random variable Y with values in $\mathcal{Y}$ and a value $y \in \mathcal{Y}$, $\mathsf{P}_{X|Y}(x, y)$ is defined as $\mathsf{P}(X = x \mid Y = y)$, that is, the probability of the event $X = x$ under the condition that $Y = y$. If a random variable X is chosen uniformly at random from a set $\mathcal{X}$, this is denoted $X \in_R \mathcal{X}$.

Algorithms and pseudo code. We specify algorithms in pseudo code, using bold-face font such as **if** to denote structural elements of the language such as branching statements and procedure headers. The notation $y \leftarrow x$ denotes that the value x is assigned to the variable y. In case this assignment is probabilistic (because, e.g., the assigned value is the output of a probabilistic function F) we write $x \leftarrow\!\!\$\ F(x)$. If an algorithm chooses a value x uniformly at random from a set $\mathcal{X}$, we write $x \leftarrow\!\!\$\ \mathcal{X}$. We generally use the symbol "$\diamond$" to denote an "error" output of an algorithm, and "$\square$" to indicate that a variable is undefined.

2.1.2 Sequences and Tuples

For two values $x, y \in \mathcal{X}$, we write $(x, y) \in \mathcal{X}^2$ for the ordered pair consisting of x and y. More generally, for $n \in \mathbb{N}$ and values $x_1, \ldots, x_n \in \mathcal{X}$, we use the notation $x^n = (x_1, \ldots, x_n) = (x_i)_{1 \le i \le n} \in \mathcal{X}^n$ to denote the finite sequence, or n-*tuple*, of values.

Strings. We use the standard notation $\mathcal{X}^*$ to denote the set of all finite sequences with arbitrary length and values in $\mathcal{X}$, formally $\mathcal{X}^* := \left(\bigcup_{n \in \mathbb{N}} \mathcal{X}^n\right) \cup \{\epsilon\}$, where ϵ denotes the unique string of length 0. We denote by $|x|$ the length of the string x, with $|\epsilon| = 0$. The concatenation of two strings is denoted by the operation "." that is defined for two strings $x = x_1 \cdots x_n$ and $x' = x'_1 \cdots x'_{n'}$ in $\mathcal{X}^*$ as $(x, x') \mapsto x.x' := x_1 \cdots x_n x'_1 \cdots x'_{n'}$.

Binary operations on bits can be extended to binary operations on bit strings $x = x_1 \cdots x_n := (x_1, \ldots, x_n) \in \{0, 1\}^*$. For instance, consider the XOR-

operation "$\oplus$," which is extended to bit strings by defining, for $x = x_1 \cdots x_n$ and $x' = x'_1 \cdots x'_n$, the operation

$$\oplus : (x, x') \quad \mapsto \quad x \oplus x' \;\coloneqq\; (x_1 \oplus x'_1) \cdots (x_n \oplus x'_n).$$

Named tuples. Formally, a pair $(x, y) \in \mathcal{X}^2$ can be considered as the set $\{(1, x), (2, y)\}$, which describes a function $[2] \to \mathcal{X}$. This justifies the alternative notation $\mathcal{X}^{[2]} = \mathcal{X}^2$. An n-tuple with values in $\mathcal{X}$ can analogously be seen as a set of pairs $(i, x_i) \in [n] \times \mathcal{X}$ which is a function in the sense that for each *label* $i \in [n]$ there is exactly one such pair in the set. More generally, we will want to consider tuples for which the labels are not elements of $[n]$ for $n \in \mathbb{N}$, but elements of arbitrary sets. Let Λ be some (discrete) label set. We write the Λ-*tuple* with elements x_λ for $\lambda \in \Lambda$ as $\langle x_\lambda \rangle_{\lambda \in \Lambda}$. We denote the set of all Λ-tuples with values in $\mathcal{X}$ as $\mathcal{X}^\Lambda$. Note that this would correspond to using the notation $\mathcal{X}^{[n]}$ for (standard) n-tuples. The notation $(x_1, \ldots, x_n) = (x_i)_{1 \le i \le n} \in \mathcal{X}^n$ is to be understood as specifying a set of pairs $\{(1, x_1), \ldots, (n, x_n)\}$ in this sense.

The tuples with labels in Λ as described above specify a value for each $\lambda \in \Lambda$. We extend the formalization to also capture "partial" tuples where some labels are not associated with a value. For such a tuple x, we write $\operatorname{supp} x$ for the *support* of x, i.e., the set of all labels $\lambda \in \Lambda$ for which a value is defined in x. We then use the notation $\mathcal{X}^{(\Lambda)}$ for the set of all *partial tuples* x with values in $\mathcal{X}$ and labels in Λ for which $\operatorname{supp} x$ is finite, formally

$$\mathcal{X}^{(\Lambda)} \quad \coloneqq \quad \bigcup_{L \subseteq \Lambda, |L| \in \mathbb{N}} \mathcal{X}^L.$$

For $x \in \mathcal{X}^{(\Lambda)}$: $\quad \operatorname{supp} x = L \iff x \in \mathcal{X}^L$.

For two partial tuples $x_1, x_2 \in \mathcal{X}^{(\Lambda)}$ with disjoint support, i.e., $\operatorname{supp} x_1 \cap \operatorname{supp} x_2 = \emptyset$, we define the joined tuple $x_1 \cup x_2$ as the union of the sets of pairs. Consequently, we use the notation $\bigcup_{i \in [n]} x_i$ for a family $x_1, \ldots, x_n$ of partial tuples with disjoint label sets. We denote by $\langle x \rangle_\lambda$ the tuple that contains only a single element x with label λ (as a set, this is $\{(\lambda, x)\}$).

Mappings on tuples. The composition of two simple functions $f : \mathcal{X} \to \mathcal{Y}$ and $g : \mathcal{Y} \to \mathcal{Z}$ is defined in the obvious way as $g \circ f : \mathcal{X} \to \mathcal{Z}, x \mapsto g(f(x))$. If the mappings are defined on tuples, i.e. $f : \mathcal{X}^\Lambda \to \mathcal{Y}$ and $g : \mathcal{Y}^\Lambda \to \mathcal{Z}$, the composition must define which argument of g is to be defined to take the output of f. Hence, for each $\bar{\lambda} \in \Lambda$ one defines a composition as

$$g \circ_{\bar{\lambda}} f : \quad \mathcal{Y}^{\Lambda \setminus \{\bar{\lambda}\}} \times \mathcal{X}^{\bar{\lambda}.\Lambda} \;\to\; \mathcal{Z}, \qquad y \cup x \;\mapsto\; g(y \cup \langle f(x) \rangle_{\bar{\lambda}}). \tag{2.1}$$

2.2 Abstract Reduction Theory

The security of most practical cryptographic schemes is based on computational assumptions. A security proof for such a scheme shows that if there is an adversary or attacker breaking the cryptographic scheme, then one can immediately translate this attacker into a solver for some computational problem. If one assumes that the computational problem is difficult to solve, then one can conclude that the cryptographic scheme is difficult to break. In complexity-theoretic terms, this describes a *reduction* from the computational problem assumed to be hard to the problem of breaking the security of the cryptographic scheme. Maurer developed an abstract reduction theory [HMS+13, Mau14] based on the concept of abstract systems as described in Section 1.1.2, which we include here for completeness.

The aim of abstract reduction theory is to describe both the computational problems and the solvers for these problems in terms of systems that are connected via their interfaces. This formulation is independent of a particular computational model and the results apply to any model that is shown to satisfy the axioms of abstract systems; one can specify computational problems and show reductions between these problems generically. The advantages of this approach are that certain reductions are described and proved abstractly without providing a concrete implementation in a computational model, and that the statements directly translate to various different types of computational models.

A computational problem is described by one or more systems and a *performance measure* that is associated with the problem and maps each solver for the problem to a performance value (often in $[0, 1]$) that describes how well the considered solver actually solves the problem. For instance, a search problem in complexity theory can be seen as a system that provides an instance of the problem to the solver, the solver is a (probabilistic) Turing machine or algorithm that obtains the instance and outputs a candidate solution, and the performance is the probability with which the solution provided by the solver is correct. The systems are connected by providing the concrete instance generated by the system describing the problem as an input to the solver. The performance measure, in the probabilistic case, is the probability that a candidate solution output by the solver is correct.

A *reduction* from a computational problem P to a computational problem P' is a function ϕ that maps each solver for the problem P' to a solver for the problem P. The quality of the reduction is measured in terms of how the performance of the solvers translates: For a function $\rho : [0, 1] \to [0, 1]$, the

function ϕ is a ρ-reduction if for all solvers S,

$$\varepsilon_{P'}(S) \quad \leq \quad \rho(\varepsilon_P(\phi(S))),$$

where ε_P is the performance function of problem P and $\varepsilon_{P'}$ is the performance function of problem P.

If the security of a cryptographic scheme is based on multiple computational problems, then a security proof corresponds to showing a (simultaneous) reduction from all these computational problems to the problem of breaking the scheme. This more general type of reduction is defined below.

Definition 2.1. Let $L \subseteq \Lambda$. Let P_λ for $\lambda \in L$ and P' be computational problems with associated performance functions ε_{P_λ} and $\varepsilon_{P'}$, respectively. Let $\rho : \prod_{\lambda \in L} [0,1] \to [0,1]$ be a monotone function. A tuple $\langle \phi_\lambda \rangle_{\lambda \in L}$ of functions mapping solver systems to solver systems is a *ρ-reduction from $\langle P_\lambda \rangle_{\lambda \in L}$ to P'* if for every solver S for P',

$$\varepsilon_{P'}(S) \qquad \leq \qquad \rho\left(\langle \varepsilon_{P_\lambda}(\phi_\lambda(S)) \rangle_{\lambda \in L}\right).$$

$\diamond$

The condition in Definition 2.1 is more compactly written as

$$\varepsilon_{P'} \qquad \leq \qquad \rho\left(\langle \varepsilon_{P_\lambda} \circ \phi_\lambda \rangle_{\lambda \in L}\right),$$

with the interpretation that a solver S is given as an argument to every function in the tuple on the right-hand side.

Composition of reductions. Two reductions are composed in the obvious way; we first describe this (for ease of notation) for the simplified setting where reductions are from a single problem to another single problem. Let P, P', and P'' be computational problems with associated performance measures ε_P, $\varepsilon_{P'}$, and $\varepsilon_{P''}$. Let $\rho, \rho' : [0,1] \to [0,1]$ and ϕ, ϕ' be mappings such that ϕ is a ρ-reduction from P to P' and ϕ' is a ρ'-reduction from P' to P''. More formally:

$$\varepsilon_{P''} \quad \leq \quad \rho' \circ \varepsilon_{P'} \circ \phi' \qquad \text{and} \qquad \varepsilon_{P'} \quad \leq \quad \rho \circ \varepsilon_P \circ \phi.$$

Then, by monotonicity, the fact that ϕ' maps systems to systems, and the associativity of function composition,

$$\varepsilon_{P''} \quad \leq \quad \rho' \circ \varepsilon_{P'} \circ \phi' \quad \leq \quad \rho' \circ (\rho \circ \varepsilon_P \circ \phi) \circ \phi' \quad \leq \quad (\rho' \circ \rho) \circ \varepsilon_P \circ (\phi \circ \phi'),$$

i.e., $\phi \circ \phi'$ is a $(\rho' \circ \rho)$-reduction from P to P''.

To extend the above statement to the general type of reduction described in Definition 2.1, we describe how the performance and solver mappings are composed.

Definition 2.2. Let $L \subseteq \Lambda$, $\langle \phi_\lambda \rangle_{\lambda \in L}$ and ρ be as in Definition 2.1. For a label $\hat{\lambda} \in L$, a set $L' \subseteq \Lambda$, and a function $\rho' : \prod_{\lambda' \in L'} [0,1] \to [0,1]$, let the tuple $\langle \phi'_\lambda \rangle_{\lambda' \in L'}$ be a ρ'-reduction. Then, the *composition of* $\langle \phi_\lambda \rangle_{\lambda \in L}$ *and* $\langle \phi'_\lambda \rangle_{\lambda' \in L'}$ *via* $\hat{\lambda}$ is defined as

$$\langle \phi_\lambda \rangle_{\lambda \in L} \circ_{\hat{\lambda}} \langle \phi'_\lambda \rangle_{\lambda' \in L'} \quad \coloneqq \quad \langle \phi_\lambda \rangle_{\lambda \in L \setminus \{\hat{\lambda}\}} \cup \langle \phi_{\hat{\lambda}} \circ \phi'_{\lambda'} \rangle_{\hat{\lambda}.\lambda' \in \hat{\lambda}.L'} \ ,$$

where $\hat{\lambda}.L' \coloneqq \left\{ \hat{\lambda}.\lambda : \lambda \in L' \right\}$. $\diamond$

The composition of the two reductions is a $(\rho \circ_{\hat{\lambda}} \rho')$-reduction, where the composition of ρ and ρ' is defined as in equation (2.1).

There are two specific types of computational problems that are relevant in the setting we consider. The first one, a *game*, formalizes the difficulty of providing a correct solution to a given problem. The second one, a *distinction problem*, formalizes that the difficulty of distinguishing between two given objects.

Abstract games. An abstract game captures a type of problem in which a solver (or game winner) is connected to a game and wins if it succeeds to provoke a certain condition. Abstract games capture search problems (the solver wins if it outputs the correct solution to the problem) or cryptographic games that capture properties such as the unforgeability of signatures (the solver wins if it forges a signature) or the hardness of inverting a trapdoor one-way permutation (the solver wins if it finds a preimage). In the deterministic case, the performance of a given winner w in a game g is either 0 (if the winner fails in winning the game) or 1 (if the winner succeeds); this is denoted as $wg = 0$ or $wg = 1$, respectively. If the game or the winner are probabilistic, the performance of the winner is defined as the probability with which the game is won.

Definition 2.3. An *abstract game* is a single-interface system G. The *game-winning performance of* W *for* G is defined as

$$[\![G]\!](W) \quad = \quad \Gamma(WG) \quad \coloneqq \quad \mathsf{P}(WG = 1),$$

for each single-interface system W. $\diamond$

The term $[\![G]\!]$ is a short notation for the performance measure that is associated to the game G, and allows to write reduction statements involving games in a compact way. In the following, we describe a (trivial) reduction which is used repeatedly in the remainder of the thesis. The lemma states that if one considers a two-interface system R that is placed between the game G

and the winner W, then one can either consider R either as being part of the winner such that WR is a winner for game G, or one can equivalently consider R as being part of the game such that W is a winner for game RG. The lemma follows immediately since we assume that the system algebra is composition-order invariant, that is, it does not matter in which order systems are composed. (In the case of the lemma, we use $(WR)G = W(RG)$.) The lemma is instrumental for proving reductions between different problems.

Lemma 2.4. *Let G be an abstract game. Then, for any two-interface system R:*

$$[\![RG]\!] \quad = \quad [\![G]\!] \circ (\cdot R),$$

where $(\cdot R)$ is considered the mapping $W \mapsto WR$.

Abstract distinction problems. The second type of problem we consider is called a distinction problem and formalizes problems where the solver has to distinguish between two objects. Formulated in terms of abstract systems, the objects are single-interface systems and connect to the solver (often called distinguisher) via this interface. Distinguishers are also single-interface systems, and the composition of a distinguisher with another single-interface system determines a bit (which can be viewed as the "output" of the distinguisher). In the deterministic case, this means that for a distinguisher d and a system s we can write $ds = 0$ or $ds = 1$. In the probabilistic case, we are interested in the difference of probability that a certain distinguisher achieves when it is connected to either of two different systems, that is, for a distinguisher D and two single-interface systems S_0 and S_1, the distinguishing advantage is defined as the absolute value of the difference in probability to output 1 when connected to S_0 or S_1, respectively.

Definition 2.5. A *distinction problem* consists of a pair (S_0, S_1) of single-interface systems and is denoted as $(\, S_0 \mid S_1)$, together with a mapping for the *distinguishing advantage* of each distinguisher D,

$$[\![(\, S_0 \mid S_1)]\!]\,(D) \quad = \quad \Delta^D\,(S_0, S_1) \quad \coloneqq \quad |\mathsf{P}(DS_0) - \mathsf{P}(DS_1)|\,.$$

$\diamond$

Note that the distinguishing advantage for a given distinguisher D can be considered a distance measure on the set of single-interface systems and satisfies the triangle inequality. More formally, for a distinguisher D and three single-interface systems S_0, S_1, and S_2, from the definition of the distinguishing advantage and the triangle inequality for the absolute value it follows that

$$\begin{aligned}
\Delta^D(S_0, S_2) &= |\mathsf{P}(DS_0 = 1) - \mathsf{P}(DS_2 = 1)| \\
&= |\mathsf{P}(DS_0 = 1) - \mathsf{P}(DS_1 = 1) + \mathsf{P}(DS_1 = 1) - \mathsf{P}(DS_2 = 1)| \\
&\leq |\mathsf{P}(DS_0 = 1) - \mathsf{P}(DS_1 = 1)| + |\mathsf{P}(DS_1 = 1) - \mathsf{P}(DS_2 = 1)| \\
&= \Delta^D(S_0, S_1) + \Delta^D(S_1, S_2).
\end{aligned}$$

Additionally, the distinguishing advantage is symmetric in the sense that for all systems S_0 and S_1:

$$\Delta^D(S_0, S_1) = \Delta^D(S_1, S_0),$$

and for all systems S it holds that $\Delta^D(S, S) = 0$. Altogether, this means that the distinguishing advantage for a fixed distinguisher D is a *pseudo metric* on the set of single-interface systems.

We prove the lemma which is is an analogue to Lemma 2.4 in the case of distinction problems. The proof is again based on the composition-order invariance of the system algebra. The lemma is again useful for proving reductions between computational problems.

Lemma 2.6. *Let $P = (S_0 \mid S_1)$ be a distinction problem. Then, for any two-interface system R:*

$$[\![(RS_0 \mid RS_1)]\!] = [\![(S_0 \mid S_1)]\!] \circ (\cdot R),$$

where $(\cdot R)$ is considered the mapping $D \mapsto DR$.

Proof. By definition and the composition-order invariance of the two-interface systems,

$$\begin{aligned}
[\![(RS_0 \mid RS_1)]\!](D) &= |\mathsf{P}(D(RS_0) = 1) - \mathsf{P}(D(RS_1) = 1)| \\
&= |\mathsf{P}((DR)S_0 = 1) - \mathsf{P}((DR)S_1 = 1)| \\
&= [\![(S_0 \mid S_1)]\!](DR) \\
&= ([\![(S_0 \mid S_1)]\!] \circ (\cdot R))(D),
\end{aligned}$$

which concludes the proof. □

2.3 Random Systems

Probabilistic discrete systems that obtain a sequence of inputs and respond to each input by providing an output are exactly captured by the concept of

random systems as introduced by Maurer [Mau02]. Within a random experiment, each input to or output from the system is a random variable with a specific probability distribution. The behavior of such a discrete system can hence be described as the conditional distribution of the outputs it provides, given the previous history, i.e., all inputs it obtained and all previously provided outputs. In this section, we recall the definition of and foundational statements about random systems from the literature [Mau02, MPR07, Mau13].

Many cryptographic primitives like block ciphers, MAC schemes, or random functions can be described as discrete systems that take a sequence $X_1, X_2, \ldots \in \mathcal{X}$ of inputs and generate, for each input $X_i \in \mathcal{X}$, an output $Y_i \in \mathcal{Y}$. Each output Y_i may depend probabilistically on all the previous inputs $X^i = (X_1, \ldots, X_i)$ as well as all the previous outputs $Y^{i-1} = (Y_1, \ldots, Y_{i-1})$. For a discrete system $\mathbf{S}$ with inputs in $\mathcal{X}$ and outputs in $\mathcal{Y}$, this is captured by a conditional probability distribution, which will be denoted by $\mathsf{p}^{\mathbf{S}}_{Y_i|X^iY^{i-1}}$ where the superscript indicates the system considered. Each conditional probability distribution involved in the definition of the random system $\mathbf{S}$ is actually a function

$$\mathsf{p}^{\mathbf{S}}_{Y_i|X^iY^{i-1}} : \quad \mathcal{Y} \times \mathcal{X}^i \times \mathcal{Y}^{i-1} \;\to\; [0,1],$$

where for all choices of the arguments x^i and y^{i-1} the sum of the function values over the choices of y_i equals 1. This motivates the formal definition of a random system.

Definition 2.7 ([Mau02])**.** An $(\mathcal{X}, \mathcal{Y})$-*random system* $\mathbf{S}$ is a (possibly infinite) sequence of conditional probability distributions $\mathsf{p}^{\mathbf{S}}_{Y_i|X^iY^{i-1}}$ for all $i \geq 1$, where $X_j \in \mathcal{X}$ and $Y_j \in \mathcal{Y}$ for all $j \geq 1$. ◇

An $(\mathcal{X}, \mathcal{Y})$-random system $\mathbf{S}$ considered in isolation does not describe a random experiment since the distribution of the inputs to the system $\mathbf{S}$ is not defined. For this reason, the conditional distributions in a random system are denoted by a lower-case letter p instead of an upper case letter P, which we use for probability distributions in a random experiment.

It is sometimes convenient to use an alternative description of a random system $\mathbf{S}$, namely the sequence of conditional distributions $\mathsf{p}^{\mathbf{S}}_{Y^i|X^i}$, where

$$\mathsf{p}^{\mathbf{S}}_{Y^i|X^i} \quad := \quad \prod_{j=1}^{i} \mathsf{p}^{\mathbf{S}}_{Y_j|X^jY^{j-1}}.$$

Note that the conditional distribution $\mathsf{p}^{\mathbf{S}}_{Y^i|X^i}$ "contains" the conditional distributions $\mathsf{p}^{\mathbf{S}}_{Y^j|X^j}$ for all $j < i$ and hence the above description of a system is redundant. The conditional distribution $\mathsf{p}^{\mathbf{S}}_{Y^i|X^i}$ must satisfy a consistency condition which ensures that Y_j for $j < i$ does not depend on $X_{j+1}, X_{j+2}, \ldots, X_i$.

Monotone binary outputs (MBOs) and games. A concept that is important in several contexts is an output of a system which is binary and monotone, that is, it is initially 0, and after potentially switching to 1 at some point, remains 1. This concept will be used for showing that systems are "almost" equivalent by specifying conditions on systems and formulating that the systems are equivalent as long as the respective MBO is 0; this allows to bound the distinction advantage by the probability of provoking the MBO to be 1. MBOs are also useful to formalized statements which are parametrized in the number of queries made to a system; this is achieved by "masking" the outputs of a system once the MBO becomes 1.

Definition 2.8 (from [MPR07])**.** For a $(\mathcal{X}, \mathcal{Y} \times \{0,1\})$-system $\mathbf{S}$ the binary component A_i of the output (Y_i, A_i) is called a *monotone binary output (MBO)* if $A_i = 1$ implies $A_j = 1$ for $j \geq i$. For such a system $\mathbf{S}$ with MBO we define two derived systems:

1. $\mathbf{S}^{-}$ is the $(\mathcal{X}, \mathcal{Y})$-system resulting from $\mathbf{S}$ by ignoring the MBO.
2. $\mathbf{S}^{\dashv}$ is the $(\mathcal{X}, \mathcal{Y} \times \{0,1\})$-system which masks the Y-output to a dummy symbol as soon as the MBO turns to 1. More precisely, the following function is applied to the outputs of $\mathbf{S}$:

$$(y, a) \mapsto (y', a) \quad \text{where} \quad y' = \begin{cases} y & \text{if } a = 0, \\ \perp & \text{if } a = 1. \end{cases}$$

$\diamond$

For a system $\mathbf{S}$ and an additional MBO $A_1, A_2, \ldots$ defined on $\mathbf{S}$, we denote the system $\mathbf{S}$ with the MBO by $\hat{\mathbf{S}}$ (where the exact MBO is usually explicitly described or clear from the context), and clearly $\hat{\mathbf{S}}^{-} = \mathbf{S}$. In general we will be interested in systems with multiple monotone binary outputs, where each output may serve a particular purpose. We will often call a system with respect to a specified MBO a *game*, and the MBO formalizes "winning the game."

Environments for random systems. Random systems can be seen as systems which only have a single interface and take inputs from the set $\mathcal{X}$ and respond with outputs in the set $\mathcal{Y}$. Random systems define a random experiment only together with an *environment* which determines the distribution of the input random variables. Intuitively, such an environment $\mathbf{E}$ for an $(\mathcal{X}, \mathcal{Y})$-system $\mathbf{S}$ corresponds to a $(\mathcal{Y}, \mathcal{X})$-system which is one query ahead: the output X_i of $\mathbf{E}$ corresponds to a query made to the random system $\mathbf{S}$, where X_i depends on the given responses Y^{i-1} of the system $\mathbf{S}$ and on the previous queries X^{i-1} made to it.

Definition 2.9 ([Mau13])**.** An $(\mathcal{X}, \mathcal{Y})$-*environment* $\mathbf{E}$ is a (possibly infinite) sequence of conditional probability distributions $\mathsf{p}^{\mathbf{E}}_{X_i|Y^{i-1}X^{i-1}}$ for $i \geq 1$, where $Y_j \in \mathcal{Y}$ and $X_j \in \mathcal{X}$ for all $j \geq 1$. $\diamond$

An $(\mathcal{X}, \mathcal{Y})$-random system $\mathbf{S}$ and an $(\mathcal{X}, \mathcal{Y})$-environment $\mathbf{E}$ together define a random experiment $\mathbf{ES}$ in which the distributions of all variables $X_1, X_2, \ldots$ and $Y_1, Y_2, \ldots$ are defined: The environment $\mathbf{E}$ defines the distribution of X_1 via $\mathsf{P}^{\mathbf{ES}}_{X_1} = \mathsf{p}^{\mathbf{E}}_{X_1}$, given the distribution of X_1 the system $\mathbf{S}$ defines the joint distribution with Y_1 via $\mathsf{P}^{\mathbf{ES}}_{X_1Y_1} = \mathsf{p}^{\mathbf{E}}_{X_1} \cdot \mathsf{p}^{\mathbf{S}}_{Y_1|X_1}$; more generally

$$\mathsf{P}^{\mathbf{ES}}_{X^iY^i} \quad = \quad \mathsf{p}^{\mathbf{E}}_{X^i|Y^{i-1}} \cdot \mathsf{p}^{\mathbf{S}}_{Y^i|X^i}.$$

Distinguishers for random systems, the distinguishing advantage, and the game winning performance. A special type of environment, namely one for which an additional binary random variable W is defined, will serve as *distinguisher* for random systems. The distinguishing advantage for a distinguisher $\mathbf{D}$ and two systems $\mathbf{S}$ and $\mathbf{T}$ is then defined as the difference in probability for this variable to be 1 between the interactions of $\mathbf{D}$ with $\mathbf{S}$ or $\mathbf{T}$.

For the random experiment to be well-defined, this means that the value of the bit W must be determined after some finite number of interactions. There are several valid ways in which this can be enforced, such as requiring the distinguisher to output the value after some fixed number of steps, or defining the system to stop after some fixed number of steps (and then defining the value as a predicate over the transcripts).

To capture a more general type of conditions than just a static number of interactions, we use a different definition. We let $\mathbf{S}$ (resp. $\mathbf{T}$) be a system with an additional "stopping" monotone binary output; while this MBO is 0, the system and the distinguisher interact normally. When the MBO switches to 1, the distinguisher determines the output bit without obtaining further outputs.[1] The restriction on a fixed number q of queries can be formalized by defining a constant MBO $A_1, A_2, \ldots, A_q = 0$ and $A_{q+1}, A_{q+2}, \cdots = 1$.

Definition 2.10. An $(\mathcal{X}, \mathcal{Y})$-*distinguisher* $\mathbf{D}$ is an $(\mathcal{X}, \mathcal{Y})$-environment which has an additional binary output W, i.e., a family of distributions $\mathsf{p}^{\mathbf{D}}_{X_i|X^{i-1}Y^{i-1}}$ and $\mathsf{p}^{\mathbf{D}}_{W_i|X^iY^{i-1}}$, where each W_i is a binary random variable. The *distinguishing advantage of* $\mathbf{D}$ *for two* $(\mathcal{X}, \mathcal{Y} \times \{0,1\})$-*systems* $\mathbf{S}$ *and* $\mathbf{T}$ (where the binary

[1] In terms of strictness, this is equivalent to explicitly blocking the outputs of the system and letting the distinguisher decide when it stops as defined with the notation $\mathbf{S}^{\dashv}$, as defined by Maurer et al. [MPR07].

component is monotone) is defined as

$$\Delta^{\mathbf{D}}(\mathbf{S},\mathbf{T}) \quad := \quad \left|\mathsf{P}^{\mathbf{DS}}(W=1) - \mathsf{P}^{\mathbf{DT}}(W=1)\right|,$$

where $\mathsf{P}^{\mathbf{DS}}(W=1)$ denotes the probability that the binary output W_q of $\mathbf{D}$ is set to 1 (and analogously with $\mathbf{T}$) if the MBO of $\mathbf{S}$ becomes 1 in step q (analogously with $\mathbf{T}$).[2] ◇

In the composition of an environment and a game, we are usually interested in the probability of the game's "winning" MBO to become 1. Recall that the game outputs one bit A_i in each step such that the sequence $A_1, A_2, \ldots$ is monotone. We define the random variable A for the MBO analogously to the random variable W above: We assume that the game has an additional "stopping" MBO that determines the particular step in which the value of the "winning" MBO is considered. If the "stopping" MBO becomes 1 in step i, then A is equal to A_i. This results in the following definition of a game winning advantage for a game winner $\mathbf{W}$ (which formally is an $(\mathcal{X},\mathcal{Y})$-environment).

Definition 2.11. For a $(\mathcal{X},\mathcal{Y}\times\{0,1\})$-random system $\mathbf{S}$ with MBO $A_1, A_2, \ldots$ and for an environment $\mathbf{W}$, we denote with $\Gamma(\mathbf{WS})$ the probability that $\mathbf{W}$ wins the game:

$$\Gamma(\mathbf{WS}) \quad := \quad \mathsf{P}^{\mathbf{WS}}(A=1).$$

◇

Relation between games and distinction problems. We describe two techniques for bounding the distinguishing advantage for the case of random systems, which are both based on the notion of game winning. If two systems behave equivalently unless a certain condition occurs, then the distinguishing advantage can be bounded by the probability of provoking that condition. In more detail, let $\mathbf{S}$ and $\mathbf{T}$ be two systems on which additionally an MBO is defined (hence, they are formally discrete games). Then these two systems are said to be *equivalent as games* if they are equivalent as long as the MBOs defined on the systems remain 0.

Definition 2.12. Two $(\mathcal{X},\mathcal{Y}\times\{0,1\})$-systems $\mathbf{S}$ and $\mathbf{T}$ with MBOs are *equivalent as games*, denoted $\mathbf{S} \overset{\mathrm{g}}{\equiv} \mathbf{T}$, if, for $i \geq 1$,

$$\mathsf{p}^{\mathbf{S}}_{Y^i,A_i=0|X^i} \quad = \quad \mathsf{p}^{\mathbf{T}}_{Y^i,A_i=0|X^i}.$$

◇

[2] Formally, this can be defined by taking the sum of probabilities for $A_q = 0 \wedge A_{q+1} = 1 \wedge W_{q+1}$ over all $q \in \mathbb{N}$.

If the two systems have the same distributions for all outputs Y_i in the case $A_i = 0$, then this implies that the probability of the event $A_i = 1$ is also the same for each game. This is formalized in the following Lemma, which is taken from Maurer [Mau13] and states that if two games are equivalent, the probability of winning is the same.

Lemma 2.13 ([Mau13, Lemma 1]). *If* $\mathbf{S} \overset{g}{\equiv} \mathbf{T}$ *for two* $(\mathcal{X}, \mathcal{Y} \times \{0,1\})$*-systems* $\mathbf{S}$ *and* $\mathbf{T}$ *with MBOs, then*

$$[\![\mathbf{S}]\!] \quad = \quad [\![\mathbf{T}]\!] \,,$$

where $[\![\mathbf{S}]\!]$ *and* $[\![\mathbf{T}]\!]$ *describe the performance in terms of provoking the MBO in the respective games.*

The following lemma then provides us with the desired bound on the distinguishing advantage: if two systems are equivalent as games, then the distinguishing advantage is upper bounded by the probability of winning the games. This lemma, which has been proven by Maurer [Mau02], is instrumental for many of our proofs.

Lemma 2.14 ([Mau13, Lemma 2]). *If* $\mathbf{S} \overset{g}{\equiv} \mathbf{T}$ *for two* $(\mathcal{X}, \mathcal{Y} \times \{0,1\})$*-systems* $\mathbf{S}$ *and* $\mathbf{T}$ *with MBOs, then*

$$[\![(\mathbf{S}^- \mid \mathbf{T}^-)]\!] \quad \leq \quad [\![\mathbf{S}]\!] \,.$$

A concept which is similar in spirit to game equivalence is that of *conditional equivalence*. A system $\mathbf{S}$ with an MBO is said to be conditionally equivalent to a system $\mathbf{T}$ without an MBO if the system $\mathbf{S}$, *conditioned on the MBO being* 0, is equivalent (as a system) to $\mathbf{T}$.

Definition 2.15. Let $\mathbf{S}$ be an $(\mathcal{X}, \mathcal{Y} \times \{0,1\})$-system with MBO and $\mathbf{T}$ be an $(\mathcal{X}, \mathcal{Y})$-system. Then $\mathbf{S}$ is *conditionally equivalent to* $\mathbf{T}$, denoted $\mathbf{S} \models\!\!\equiv \mathbf{T}$, if, for $i \geq 1$,

$$\mathsf{p}^{\mathbf{S}}_{Y^i|X^i,A_i=0} \quad = \quad \mathsf{p}^{\mathbf{T}}_{Y^i|X^i} \,,$$

where the equality holds on all arguments for which both terms are defined. $\diamond$

The most important application for conditional equivalence, as for game equivalence, is to bound the advantage in distinguishing two systems. In contrast to Lemma 2.14, however, the advantage is even bounded by the probability of provoking the condition "blindly," i.e., independently of the outputs of the system. A consequence of this theorem is that the best distinguisher for $\mathbf{S}$ and $\mathbf{T}$ is non-adaptive.

Theorem 2.16 ([Mau13, Theorem 3]). *If for $(\mathcal{X}, \mathcal{Y})$-systems $\mathbf{S}$ and $\mathbf{T}$ one can define an MBO $A_1, A_2, \ldots$ such that $\hat{\mathbf{S}} \models \mathbf{T}$, then*

$$[\![(\mathbf{S} \mid \mathbf{T})]\!] \quad \leq \quad \left[\!\!\left[\vec{\mathbf{T}}\hat{\mathbf{S}}\right]\!\!\right],$$

where $\vec{\mathbf{T}}$ is the system that answers the queries (as $\mathbf{T}$) at the left interface and forwards the same queries to the right interface.[3]

2.4 Cryptographic Schemes and Security Properties

In the literature, a cryptographic scheme us usually described as a tuple of algorithms, and a security definition in the context of secure communication formalizes a property of such a scheme. Each property is then defined by a game which involves the scheme, and the scheme is said to have the property if no efficient adversary can win the game. In this section, we formalize several notions from the cryptographic literature in a way that is compatible with the remainder of the thesis.

2.4.1 Game-based Definitions

In the literature, game-based definitions specify a property of a cryptographic scheme based on an interaction between two (hypothetical) entities: the game (or challenger) and the adversary. During this interaction, the adversary issues certain "oracle queries" to the challenger; these queries model the use of the scheme in applications. The adversary's goal is specified by the game, and could be, e.g., forging a message or extracting useful information about the plaintext from a ciphertext. If this game cannot be won by any (efficient) adversary, then the scheme is secure against the considered type of attack. The adversary is often formalized as a Turing machine that has access to a set of oracles formalizing the challenger.

The literature often distinguishes between two types of "games." The first one captures that a certain task be infeasible, such as forging a signature or inverting a one-way function. This type of game corresponds to a random system with an MBO as described in Section 2.3, and the performance of the adversary is measured similarly to Definition 2.11. The second type of game formalizes the problem of distinguishing between two scenarios, such

[3]For a fixed distinguisher $\mathbf{D}$, the notation $[\![\mathbf{DT}]\!]$ is used for the system $\mathbf{D}\vec{\mathbf{T}}$ by Maurer [Mau13] and defined in [Mau13, Definition 7]. We chose a different notation because $\vec{\mathbf{T}}$ can be considered a system that is defined even without connecting a distinguisher to it.

as distinguishing between ciphertexts for different messages. This type of problem corresponds, in our model, to a distinction problem in the spirit of Definition 2.10.

Since the majority of the existing literature considers "closed classes" of adversaries, such as algorithms that run in polynomial time, it makes sense to state and prove that a scheme "has a certain property." In our reductionist view on security, this is not possible, and we describe the properties by the problem that is supposed to be solved to break the property, and the associated performance measure. A security proof is then stated as an explicit reduction that maps every distinguisher from the random experiment defining a construction to a solver/adversary in the corresponding game. In this view, restrictions on the adversary, such as the maximum number of queries that can be used to win a game, are a property of the game, not the solver/adversary that tries to win the game. This restriction is formalized, as described in Section 2.3, as a special MBO that becomes 1 once the number of queries exceeds the specified bound. The obtained definition is formally equivalent to the standard formulation; however, it integrates smoothly with the type of security statement we are interested in.

In the remainder of the section, we describe security notions from the literature by pseudo-code descriptions of random systems, i.e., sequences of conditional distributions $\mathsf{p}_{X_i|X^iY^{i-1}}$. The description is to be interpreted as follows. Upon the first input X_1, the game initially evaluates the described **init** procedure. The input X_1, if non-empty, is then interpreted as a procedure call with corresponding parameters (using some fixed encoding of the procedure name and the parameter list), and the system returns as output Y_1 the value prescribed by the procedure (or the output of **init** if the input was empty). All following inputs $X_2, X_3, \ldots$ are processed analogously, but without the evaluation of the **init** procedure. For games that are formalized with an MBO, there is a specific variable W and setting $W \leftarrow 1$ in a procedure effects that the MBO is set.

2.4.2 Message Authentication Codes

The purpose of a message authentication code (MAC) is to authenticate message transmissions in a setting where two parties already share a secret key. MACs are often deterministic, i.e., one computes a function on the secret key and the message that is to be authenticated. The computed value, often called *tag*, is then sent along with the message, and the receiver can check, again using the secret key, whether a received pair of message and tag is valid.

Definition 2.17. A *MAC scheme* with key space[4] $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$ is a pair of functions $\mathrm{MAC} = (\mathrm{mac}, \mathrm{check})$, such that $\mathrm{mac} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ and $\mathrm{check} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{\underline{\mathsf{true}}, \underline{\mathsf{false}}\}$. A MAC is *correct* if for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$, $\mathrm{check}(k, m, \mathrm{mac}(k, m)) = \underline{\mathsf{true}}$. ◇

A MAC scheme is secure if, intuitively, no adversary can produce valid message-tag pairs, and this intuition is usually captured in a game [BCK96]. There are two different flavors of MAC security.

Weak unforgeability under chosen-message attack (WUF-CMA). Weak unforgeability, intuitively, means that no (efficient) adversary, even with access to an oracle that produces valid tags for chosen messages, will be able to generate a valid tag for a message for which it did not obtain a valid tag from the oracle before. System 1 describes the weak unforgeability game. The $\mathbf{tag}$-oracle allows the adversary to obtain valid tags for chosen messages, and the $\mathbf{vrf}$-oracle allows to check for the validity of tags. As pointed out by Bellare et al. [BGM04], it is important to allow the adversary multiple verification queries.

System 1 WUF-CMA Game $\mathbf{G}^{\text{WUF-CMA}}(\mathrm{MAC})$

```
procedure init
    k ←$ K
    B ← ∅
end procedure

procedure tag(m)
    t ← mac(k, m)
    B ← B ∪ {m}
    return t
end procedure

procedure vrf(m, t)
    b ← check(k, m, t)
    W ← W ∨ (b ∧ (m ∉ B))
    return b
end procedure
```

The weak unforgeability game for a MAC scheme MAC together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to one of the $\mathbf{tag}$ or $\mathbf{vrf}$ oracles is denoted as $\mathbf{G}_q^{\text{WUF-CMA}}(\mathrm{MAC})$, and the performance of an adversary in breaking MAC within q queries is defined by $[\![\mathbf{G}_q^{\text{WUF-CMA}}(\mathrm{MAC})]\!]$.

Strong unforgeability under chosen-message attack (SUF-CMA). Strong unforgeability, intuitively, means that no (efficient) adversary, even with ac-

[4] Usually, $\mathcal{M} = \{0,1\}^*$, $\mathcal{K} = \{0,1\}^k$ for some $k \in \mathbb{N}$, and $\mathcal{T} = \{0,1\}^t$ for some $t \in \mathbb{N}$.

cess to an oracle that produces valid tags for chosen messages, will be able to generate a valid message-tag pair other than the pairs obtained from the oracle. The difference to weak unforgeability is that if there was a query $\mathbf{tag}(m)$ with a response t, then strong unforgeability means that (m, t') for $t \neq t' \in \mathcal{T}$ with $\mathrm{check}(k, m, t') = \underline{\mathsf{true}}$ would be considered a forgery, whereas this is not the case for weak unforgeability. System 2 describes the strong unforgeability game.

System 2 SUF-CMA Game $\mathbf{G}^{\text{SUF-CMA}}(\mathrm{MAC})$

```
procedure init
    k ←$ K
    B ← ∅
end procedure

procedure tag(m)
    t ← mac(k, m)
    B ← B ∪ {(m, t)}
    return t
end procedure

procedure vrf(m, t)
    b ← check(k, m, t)
    W ← W ∨ (b ∧ ((m, t) ∉ B))
    return b
end procedure
```

The strong unforgeability game for a MAC scheme MAC together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to one of the $\mathbf{tag}$ or $\mathbf{vrf}$ oracles is denoted as $\mathbf{G}_q^{\text{SUF-CMA}}(\mathrm{MAC})$, and the performance of an adversary in breaking MAC within q queries is defined by $[\![\mathbf{G}_q^{\text{SUF-CMA}}(\mathrm{MAC})]\!]$.

2.4.3 Symmetric Encryption

The purpose of a symmetric encryption scheme is to protect the confidentiality of plaintext messages in a setting where two parties already share a secret key. The sender computes an (often probabilistic) function on the plaintext and the secret key, and sends the obtained ciphertext to the receiver. The receiver then applies the (usually deterministic) decryption function to the ciphertext and the secret key to recover the original plaintext.

Definition 2.18. An *encryption scheme* with plaintext space $\mathcal{M}$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$ is a pair $\mathrm{SC} = (\mathrm{enc}, \mathrm{dec})$ of (possibly probabilistic) functions $\mathrm{enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and $\mathrm{dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\diamond\}$. An encryption scheme is *correct* if for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$, $\mathrm{dec}(k, \mathrm{enc}(k, m)) = m$. $\diamond$

The fundamental goal of an encryption scheme is to protect the confidentiality of the transmitted messages. The standard security notions for confi-

dentiality are IND-CPA and IND-CCA, i.e., indistinguishability (of ciphertexts) under chosen-plaintext and chosen-ciphertext attack, respectively. Several variants appear in the literature; in all variants, a bit $B \in \{0,1\}$ is chosen uniformly at random, and, depending on the variant, the adversary has access to one of the following settings of oracles:

- multiple queries at a "real-or-random" oracle where, in each query, the adversary inputs a plaintext m_0, the game chooses m_1 with $|m_0| = |m_1|$ uniformly at random, and returns an encryption of m_B;
- multiple queries at a "left-or-right" oracle where the adversary inputs two messages m_0 and m_1 with $|m_0| = |m_1|$ and obtains an encryption of m_B;
- multiple queries at an "encryption" oracle where, on input m, the adversary obtains an encryption of m, as well as one "real-or-random" query;
- multiple "encryption" queries and one "left-or-right" query.

Finally, the adversary has to guess the bit B (with probability non-negligibly different from $1/2$). It turns out that, for any encryption scheme, the advantages that can be achieved in the above games are related by a factor that is either a constant or linear in the number of queries [BDJR97].

IND-CPA. The term IND-CPA usually refers to a game where the adversary has access to the oracles described in one of the four settings above. In this thesis, we use the "real-or-random" definition of IND-CPA, because it relates to constructive security definitions in a clear way. Relations to the other notions follow by the statements shown by Bellare et al. [BDJR97].

System 3 IND-CPA System $\mathbf{G}_b^{\text{IND-CPA}}(\mathrm{SC})$

```
procedure init
    k ←$ K
end procedure

procedure enc(m_0)
    m_1 ←$ {0,1}^|m_0|
    c ←$ enc(k, m_b)
    return c
end procedure
```

The systems describing the IND-CPA game for an encryption scheme SC together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to the **enc** oracle are denoted as $\mathbf{G}_{q,b}^{\text{IND-CPA}}(\mathrm{SC})$ for $b \in \{0,1\}$, and the performance of an adversary in breaking SC within q queries is defined by $[\![(\mathbf{G}_{q,0}^{\text{IND-CPA}}(\mathrm{SC}) \mid \mathbf{G}_{q,1}^{\text{IND-CPA}}(\mathrm{SC}))]\!]$.

IND-CCA. In the IND-CCA game, the adversary is, in addition to one type of oracles of the IND-CPA game, given access to a decryption oracle where it can query ciphertexts that are different from those he obtained from the encryption oracle. The reason for the latter restriction is that if the adversary were allowed to decrypt the ciphertexts obtained in the game, winning the game would become trivial.

System 4 IND-CCA System $\mathbf{G}_b^{\text{IND-CCA}}(\mathrm{SC})$

```
procedure init
    k ←$ K
    B ← ∅
end procedure

procedure enc(m_0)
    m_1 ←$ {0,1}^|m_0|
    c ←$ enc(k, m_b)
    B ← B ∪ {c}
    return c
end procedure

procedure dec(c)
    if (c ∉ B) then
        m ← dec(k, c)
        return m
    end if
end procedure
```

The systems in the IND-CCA game for an encryption scheme SC together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to either the $\mathbf{enc}$ or $\mathbf{dec}$ oracles are denoted as $\mathbf{G}_{q,b}^{\text{IND-CCA}}(\mathrm{SC})$ for $b \in \{0,1\}$, and the performance of an adversary in breaking SC within q queries is defined by $[\![(\mathbf{G}_{q,0}^{\text{IND-CCA}}(\mathrm{SC}) \mid \mathbf{G}_{q,1}^{\text{IND-CCA}}(\mathrm{SC}))]\!]$.

INT-CTXT. Integrity of ciphertexts has been introduced by Bellare and Namprempre [BN00], and formalizes that the adversary cannot produce *any* fresh valid ciphertext. In more detail, an encryption scheme is said to achieve INT-CTXT security if no adversary with access to an encryption oracle can generate a valid ciphertext that is different from all ciphertexts obtained from the oracle. Here, "valid" means that the decryption outputs a message (not an error symbol). Note that existential unforgeability [KY00b] and ciphertext unforgeability [Kra01] are similar: The differences are, for example, that the definition of Bellare and Namprempre [BN00] allows multiple queries to the challenge oracle, whereas the one of Katz and Yung [KY00b] allows only one.

The system in the INT-CTXT game for an encryption scheme SC together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to either the $\mathbf{enc}$ or $\mathbf{check}$ oracles is denoted as $\mathbf{G}_q^{\text{IND-CTXT}}(\mathrm{SC})$, and the performance of an adversary in breaking SC within q queries is defined by $[\![\mathbf{G}_q^{\text{IND-CTXT}}(\mathrm{SC})]\!]$.

System 5 INT-CTXT Game $\mathbf{G}^{\text{INT-CTXT}}(\text{SC})$

```
procedure init
    k ←$ K
    B ← ∅
end procedure

procedure enc(m)
    c ← enc(k, m)
    B ← B ∪ {c}
    return c
end procedure

procedure check(c)
    b ← (dec(k, c) ≠ ◇)
    W ← W ∨ (b ∧ (c ∉ B))
    return b
end procedure
```

2.4.4 Public-Key Encryption

The purpose of a public-key encryption (PKE) scheme is to allow a sender to encrypt messages for a receiver in a setting without a pre-shared secret key. The receiver generates a key pair consisting of a public and a private (or secret) key, and publishes the public key. The encryption of a message is computed based on the public key, but only the receiver (who knows the private key) can decrypt ciphertexts.

Definition 2.19. A *public-key encryption (PKE) scheme* with message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, public-key space $\mathcal{PK}$, and secret-key space $\mathcal{SK}$ consists of three (possibly probabilistic) functions $\text{PKE} = (\text{PKEgen}, \text{PKEenc}, \text{PKEdec})$: a (probabilistic) key-generation function $\text{PKEgen} : \emptyset \to \mathcal{PK} \times \mathcal{SK}$, a (probabilistic) encryption function $\text{PKEenc} : \mathcal{PK} \times \mathcal{M} \to \mathcal{C}$, and an (often deterministic) decryption function $\text{PKEdec} : \mathcal{SK} \times \mathcal{C} \to \mathcal{M} \cup \{\diamond\}$. A PKE scheme is *correct* if for all key pairs $(pk, sk) \in \text{im}(\text{PKEgen}) \subseteq \mathcal{PK} \times \mathcal{SK}$ and $m \in \mathcal{M}$, $\text{PKEdec}(sk, \text{PKEenc}(pk, m)) = m$. $\diamond$

The key-generation function PKEgen outputs a pair $(pk, sk) \leftarrow\!\!\$\ \text{PKEgen}()$ of keys, the encryption function PKEenc takes a message $m \in \mathcal{M}$ and a public key $pk \in \mathcal{K}$ and outputs a ciphertext $c \leftarrow\!\!\$\ \text{PKEenc}(pk, m)$, and the (usually deterministic) decryption function PKEdec takes a ciphertext $c \in \mathcal{C}$ and a secret key sk and outputs a plaintext $m \leftarrow \text{PKEdec}(sk, c)$. The decryption algorithm may output the special symbol $\diamond$, indicating that the ciphertext c is invalid. The most important properties for our work are confidentiality, key privacy, and robustness.

Confidentiality. The most commonly required property of PKE schemes is indistinguishability under chosen-ciphertext attacks (IND-CCA). Usually, IND-

CCA is defined as a left-or-right (LoR) indistinguishability game, where an adversary must guess which of two messages m_0 and m_1 of his choice are encrypted under a known public key, depending on a hidden bit B. The game is formalized as $\mathbf{G}_b^{\text{PKE-CCA}}(\text{PKE})$ in System 6, where the name is chosen to avoid a collision with the IND-CCA game for symmetric encryption.

System 6 IND-CCA System $\mathbf{G}_b^{\text{PKE-CCA}}(\text{PKE})$

```
procedure init
    (sk, pk) ←$ PKEgen()
    cc ← □
    return pk
end procedure

procedure dec(c)
    if c ≠ cc then
        m ← PKEdec(sk, c)
        return m
    end if
end procedure

procedure chgen(m_0, m_1)
    if cc = □ then
        cc ←$ PKEenc(pk, m_b)
        return cc
    end if
end procedure
```

The systems in the IND-CCA game for a PKE scheme PKE together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to the $\mathbf{dec}$ oracle are denoted as $\mathbf{G}_{q,b}^{\text{PKE-CCA}}(\text{PKE})$ for $b \in \{0,1\}$, and the performance of an adversary in breaking PKE within q queries is defined by

$$[\![\big(\mathbf{G}_{q,0}^{\text{PKE-CCA}}(\text{PKE}) \mid \mathbf{G}_{q,1}^{\text{PKE-CCA}}(\text{PKE}) \big)]\!] \,.$$

Key privacy. In a key-private PKE scheme, the adversary, given two public keys pk_0 and pk_1, must be unable to tell which key was used to generate a given ciphertext [BBDP01]. The purpose of this definition is to formalize that an adversary cannot determine *with respect to which public key* an encryption was computed, hence, this is supposed to formalize an anonymity guarantee for the receiver. The definition is similar in spirit to the standard "left-or-right" IND-CCA definition, where the adversary is given the public key, but does not know which of two messages is encrypted under it. In the key-privacy game the message is known, but not the public key. The notion can be formalized as a distinction problem between two systems $\mathbf{G}_b^{\text{IK-CCA}}$ (for $b \in \{0,1\}$), which is described as System 7.

In particular, we denote by cc the challenge ciphertext. The decryption oracle $\mathbf{dec}(\cdot,\cdot)$ takes as input a ciphertext c and a bit $b \in \{0,1\}$, the latter indicating under which secret key c shall be decrypted. If the adversary asks to

decrypt the challenge ciphertext *after* the challenge has been generated, the algorithm returns without value; else, it returns the decryption of the ciphertext c under sk_b. The challenge generation algorithm **chgen** can be invoked only exactly once during the game. (If the adversary runs this procedure for a second time, the game remains inactive.) The challenge ciphertext is then output to the adversary.

System 7 IK-CCA System $\mathbf{G}_{q,b}^{\text{IK-CCA}}(\mathrm{PKE})$

```
procedure init
    (sk_0, pk_0) ←$ PKEgen()
    (sk_1, pk_1) ←$ PKEgen()
    cc ← □
    return (pk_0, pk_1)
end procedure

procedure dec(c, b')
    if c ≠ cc then
        m ← PKEdec(sk_b', c)
        return m
    end if
end procedure

procedure chgen(m)
    if cc = □ then
        cc ←$ PKEenc(pk_b, m)
        return cc
    end if
end procedure
```

The systems in the IK-CCA game for a PKE scheme PKE together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to the **dec** oracle are denoted as $\mathbf{G}_{q,b}^{\text{IK-CCA}}(\mathrm{PKE})$ for $b \in \{0,1\}$, and the performance of an adversary in breaking PKE within q queries is defined by

$$[\![(\mathbf{G}_{q,0}^{\text{IK-CCA}}(\mathrm{PKE}) \mid \mathbf{G}_{q,1}^{\text{IK-CCA}}(\mathrm{PKE}))]\!] .$$

Robustness. The notion of *robustness* in encryption was formalized by Abdalla et al. [ABN10] to specify the behavior of an encryption scheme on ciphertexts that were generated with respect to different public keys. The notion comes in two flavors: *weak* and *strong* robustness, both with versions under chosen-plaintext and under chosen-ciphertext attacks. We focus here on weak robustness under chosen-ciphertext attacks (WROB-CCA), associated with the game described as System 8, where the adversary may call the following oracles.

- On input an identifier id, the oracle $\mathbf{genuser}(\cdot)$ generates a public and a private key for the user id and returns the public key. A set $\mathcal{U}$ keeps track of the users generated by the $\mathbf{genuser}(\cdot)$ oracle, i.e. the honestly generated key pairs.

- On input a valid identifier $id \in \mathcal{U}$, the oracle $\mathbf{corrupt}(\cdot)$ returns the private key corresponding to user id and adds the identifier to a set $\mathcal{V}$.
- On input a valid identifier $id \in \mathcal{U}$ and a ciphertext c, the decryption oracle $\mathbf{dec}(\cdot,\cdot)$ outputs the corresponding plaintext m.

We modify the original game in the sense that instead of a **finalize** oracle that allows to specify a pair of identities (id_0, id_1) of uncorrupted identities, encrypts a message under id_0 and decrypts under id_1, we describe a **enc** oracle that allows to specify the identity for encrypting and decrypts under all other uncorrupted identities. Additionally, this oracle can be queried multiple times, and returns the obtained ciphertext c. By a standard "guessing" argument, for n users and up to q messages, this can be seen to relate to the original definition by a factor of nq. The symbol "$*$" in the description of the following game is to be understood as a wildcard.

System 8 WROB-CCA Game $\mathbf{G}^{\text{WROB-CCA}}(\text{PKE})$

```
procedure init
    U, V ← ∅
end procedure

procedure genuser(id)
    (sk_id, pk_id) ←$ PKEgen()
    U ← U ∪ {(id, sk_id, pk_id)}
    return pk_id
end procedure

procedure corrupt(id)
    if (id, *, *) ∈ U then
        V ← V ∪ {id}
        return sk_id from U
    end if
end procedure

procedure enc(m, id_0)
    c ←$ PKEnc(pk_{id_0}, m)
    for id ∈ U \ (V ∪ {id_0}) do
        m_id ← PKEdec(sk_id, c)
        W ← W ∨ (m_id ≠ ◇)
    end for
    return c
end procedure

procedure dec(id, c)
    if (id, *, *) ∈ U then
        m ← PKEdec(sk_id, c)
        return m
    end if
end procedure
```

The system in the WROB-CCA game for a PKE scheme PKE in a setting with n users together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to the **dec** or **check** oracles or more than $n \in \mathbb{N}$ queries have been made to the **genuser**-oracle is denoted as $\mathbf{G}^{\text{WROB-CCA}}_{n,q}(\text{PKE})$, and the performance of an adversary in breaking the weak robustness of PKE within q queries and with n users is defined by $[\![\mathbf{G}^{\text{WROB-CCA}}_{n,q}(\text{PKE})]\!]$.

2.4.5 Key-Encapsulation Mechanisms

The purpose of a key-encapsulation mechanism (KEM) is to allow a sender to transmit a secret key to a receiver in the same setting as for public-key encryption. The receiver generates a key pair consisting of a public and a private (or secret) key, and publishes the public key. The encapsulation of a key is computed based only on the public key, but only the receiver (who knows the private key) can decapsulate the key from a ciphertext.

Definition 2.20. A *key-encapsulation mechanism (KEM)* with key space $\mathcal{K}$, ciphertext space $\mathcal{C}$, public-key space $\mathcal{PK}$, and secret-key space $\mathcal{SK}$ consists of three possibly probabilistic functions $\mathrm{KEM} = (\mathrm{KEMgen}, \mathrm{KEMenc}, \mathrm{KEMdec})$: a (probabilistic) key-generation function $\mathrm{KEMgen} : \emptyset \to \mathcal{PK} \times \mathcal{SK}$, a (probabilistic) encapsulation function $\mathrm{KEMenc} : \mathcal{PK} \to \mathcal{K} \times \mathcal{C}$, and an (often deterministic) decapsulation function $\mathrm{KEMdec} : \mathcal{SK} \times \mathcal{C} \to \mathcal{K} \cup \{\diamond\}$. A KEM is *correct* if for all key pairs $(pk, sk) \in \mathrm{im}(\mathrm{PKEgen}) \subseteq \mathcal{PK} \times \mathcal{SK}$ and $(k, c) \in \mathrm{im}(\mathrm{KEMenc}(pk))$, $\mathrm{KEMdec}(sk, c) = k$. $\diamond$

The (probabilistic) key-generation function $\mathrm{KEMgen} : \emptyset \to \mathcal{PK} \times \mathcal{SK}$ outputs a key pair (pk, sk), the (probabilistic) encapsulation function $\mathrm{KEMenc} : \mathcal{PK} \to \mathcal{K} \times \mathcal{C}$ takes a public key $pk \in \mathcal{K}$ and outputs a pair of key and ciphertext $(k, c) \leftarrow\$ \mathrm{KEMenc}(pk)$, and the (usually deterministic) decapsulation function $\mathrm{KEMdec} : \mathcal{SK} \times \mathcal{C} \to \mathcal{K} \cup \{\diamond\}$ takes a secret key sk and a ciphertext $c \in \mathcal{C}$ and outputs a key $k \leftarrow \mathrm{KEMdec}(sk, c)$. The decapsulation function may output the special symbol $\diamond$; this indicates that the ciphertext c is invalid.

The most important property for our work is indistinguishability of the encapsulated key from an independent and uniformly random one. This property is, analogously to the public-key encryption case, often referred to as IND-CPA. The game is formalized as $\mathbf{G}_b^{\text{KEM-CPA}}$ in System 9, where the name is chosen to avoid a collision with the IND-CPA game for symmetric encryption.

System 9 IND-CPA System $\mathbf{G}_b^{\text{KEM-CPA}}(\mathrm{KEM})$

```
procedure init
    (sk, pk) ←$ KEMgen()
    return pk
end procedure

procedure chgen
    (k_0, c) ←$ KEMenc(pk)
    k_1 ←$ K
    return (k_b, c)
end procedure
```

The systems in the IND-CPA game for a KEM scheme KEM are denoted as $\mathbf{G}_b^{\text{KEM-CPA}}(\mathrm{KEM})$ for $b \in \{0, 1\}$, and the performance of an adversary in

breaking KEM is defined by

$$[\![(\mathbf{G}_0^{\text{KEM-CPA}}(\mathrm{KEM}) \mid \mathbf{G}_1^{\text{KEM-CPA}}(\mathrm{KEM}))]\!] .$$

2.4.6 Signatures

The purpose of a signature scheme is to allow a sender to authenticate messages for a receiver in a setting without a pre-shared secret key. The sender initially generates a key consisting of a (private) signature key and a (public) verification key, and publishes the verification key. The signature of a message is computed based on the private signature key and can only be computed by the party possessing that key, but signatures can be verified by any party possessing the verification key.

Definition 2.21. A *signature scheme* with message space $\mathcal{M}$, signature space $\mathcal{S}$, signature-key space $\mathcal{SK}$, and verification-key space $\mathcal{VK}$ consists of three (possibly probabilistic) functions $\mathrm{SIG} = (\mathrm{SIGgen}, \mathrm{SIGsgn}, \mathrm{SIGvrf})$: a (probabilistic) key-generation function $\mathrm{SIGgen} : \emptyset \to \mathcal{SK} \times \mathcal{VK}$, a (probabilistic) signing function $\mathrm{SIGsgn} : \mathcal{SK} \times \mathcal{M} \to \mathcal{S}$, and an (often deterministic) verification function $\mathrm{SIGvrf} : \mathcal{SK} \times \mathcal{M} \times \mathcal{S} \to \{\underline{\mathsf{true}}, \underline{\mathsf{false}}\}$. A signature scheme is *correct* if for all key pairs $(sk, vk) \in \mathrm{im}(\mathrm{SIGgen}) \subseteq \mathcal{SK} \times \mathcal{VK}$ and $m \in \mathcal{M}$, $\mathrm{SIGvrf}(vk, m, \mathrm{SIGsgn}(sk, m)) = \underline{\mathsf{true}}$. ◇

The *key-generation* function SIGgen takes no input and outputs a pair (sk, vk) of a *signature key* sk and a *verification key* vk. The *signing* function SIGsgn takes as input a signature key sk and a message $m \in \{0,1\}^*$, and outputs a signature $s \leftarrow_{\$} \mathrm{SIGsgn}(sk, m)$. The (often deterministic) *verification* function SIGvrf takes as input a verification key vk, a message m, and a signature s, and outputs a decision value. The common security requirement for a signature scheme SIG is called *unforgeability* and is formalized by the game $\mathbf{G}^{\text{EUF-CMA}}(\mathrm{SIG})$. This corresponds to WUF-CMA for MACs.

The existential unforgeability game for a signature scheme SIG together with an MBO that becomes 1 as soon as more than $q \in \mathbb{N}$ queries have been made to one of the **sign** or **vrf** oracles is denoted as $\mathbf{G}_q^{\text{EUF-CMA}}(\mathrm{SIG})$, and the performance of an adversary in breaking SIG within q queries is defined by $[\![\mathbf{G}_q^{\text{EUF-CMA}}(\mathrm{SIG})]\!]$.

System 10 EUF-CMA Game $\mathbf{G}^{\text{EUF-CMA}}(\text{SIG})$

```
procedure init
    (sk, vk) ←$ SIGgen
    B ← ∅
    return vk
end procedure

procedure sign(m)
    s ←$ SIGsgn (sk, m)
    B ← B ∪ {m}
    return t
end procedure

procedure vrf(m, s)
    b ← SIGvrf (vk, m, s)
    W ← W ∨ (b ∧ (m ∉ B))
    return b
end procedure
```

Chapter 3

A Framework for Constructive Cryptography

This chapter introduces a formal framework that instantiates the concepts of abstract and constructive cryptography introduced by Maurer and Renner [MR11]. It follows the top-down approach of abstract cryptography, starting with the formalization of the abstract paradigm underlying the security statements and step-by-step refining the concepts while, for each additional layer, proving that the axioms of the previously described layer are satisfied.

In Section 3.1, we describe the notion of *construction* from Maurer and Renner [MR11], which can be seen as the characteristic feature of constructive cryptography. We also define what it means for a construction to be composable. Composability can be understood as the property that the guarantee formalized by a construction notion is useful in a larger context (i.e., for higher-level protocols), and allows complex protocols to be built from smaller sub-protocols in a modular fashion.

We continue in Section 3.2 by providing an algebraic description of the *abstract systems* concept of Maurer and Renner [MR11]. On a high level, abstract systems capture the topology in which multiple interacting systems are connected. They allow to formulate and prove certain statements, such as the composition theorem, by only manipulating symbolic expressions. We define the construction notion for cryptographic protocols in the setting with a single (external) attacker, in Section 3.3, based on the formalism of abstract systems. We also discuss how parametrized constructions are defined, which allow for making statements that depend on, for instance, a security parameter such as the key length or the set of statically corrupted parties.

Finally, Section 3.4.1 contains a formalization of *discrete systems*, which

allows to actually describe the behavior of cryptographic protocols and to prove their security. The particular notion we describe here formalizes behavior that is described by a distribution over monotone functions, and we show that systems of this type satisfy the axioms of the abstract systems algebra described in Section 3.2. We developed this formalism independently by asking for the minimal type of discrete behavior in the context of system algebras; we later found out that it can be seen as a probabilistic version of the monotone functions in Kahn networks [Kah74]. In Section 3.4.2, we describe how statements can be parametrized to capture bounds based on "dynamic" parameters like the number or length of inputs. Section 3.4.3 explains how the reduction proofs in the subsequent chapters are performed, and Section 3.4.4 introduces a specification language for discrete systems.

3.1 The Construction Concept

A central and well-known paradigm in constructive disciplines is to construct a complex system from simpler component systems or modules, which each may consist of yet simpler component systems, and so on. Each such construction step is described by a *constructor* that specifies how the given components (elements of some *component set* Ω) are combined to construct a more useful or otherwise desirable component (again in the set Ω); a constructor hence formalizes a certain method or scheme that combines and transforms the components. The sequential application of several construction steps can again be seen as a single construction step. This is formalized by a composition operation "$\circ$" on the set of constructors that allows to "chain" multiple construction steps.

Definition 3.1. A *constructor set* Γ is a set equipped with a (partial) composition operation $\circ : \Gamma \times \Gamma \to \Gamma$. $\diamond$

A *construction notion* can be seen as a relation in $\Omega \times \Gamma \times \Omega$, where the statement that a triple $(R, a, S) \in \Omega \times \Gamma \times \Omega$ is in that relation means that the constructor a *assumes* the component R and *constructs* from it the component S. The definition of the considered relation determines which constructions are valid.

Definition 3.2. A *construction notion* for a component set Ω and a constructor set Γ is a subset of $\Omega \times \Gamma \times \Omega$. A construction is often denoted by an arrow, for example $\longrightarrow$, as follows: If (R, a, S) is in the construction, then we write $R \xrightarrow{a} S$ and say that the component S *is constructed from* R *by* a. $\diamond$

A construction notion is only useful if the constructed components can be used as assumed components in subsequent construction steps. This prop-

erty of a construction notion is called *composability* and is formalized in the following definition. It corresponds to the standard notion of transitivity for binary relations.

Definition 3.3. A construction $\longrightarrow$ for component set Ω and constructor set Γ is called *serially composable* if the following property holds for all $R, S, T \in \Omega$ and $a, b \in \Gamma$:

$$R \xrightarrow{a} S \quad \wedge \quad S \xrightarrow{b} T \quad \Longrightarrow \quad R \xrightarrow{b \circ a} T.$$

$\diamond$

The composability of a construction assures that multiple construction steps can be applied subsequently. This is instrumental for a clean and modular protocol design.

The original definition of Maurer and Renner [MR11] requires the existence of a constructor id that behaves as the identity on the components:

$$\forall R \in \Omega : \qquad R \xrightarrow{\mathsf{id}} R.$$

Furthermore, the construction is required to be *context insensitive* in the sense that the a constructor a that constructs S from R still applies if additional components are available in parallel to R and S. We do not require these two properties. The identity constructor described by Maurer and Renner [MR11] is required because of the way in which they define parallel composition; it would be redundant in our formulation.[1] Context insensitivity does not hold for the construction notion we consider in this thesis. The reason is that the definition of construction is based on explicit reductions, and the context indeed affects the reduction. While the constructions are not context insensitive, the influence of the context can be specified exactly, and Lemma 3.14 describes this effect.

3.2 Abstract Systems

In settings where multiple objects are connected, one needs a formalism to describe the topology of the composite object. This holds, for instance, for computer networks where the topology describes which computers communicate, but also for a setting with multiple interactive systems (like algorithms

[1] Roughly, when "embedding" a construction in a context, the identity construction assures that the context can be left unchanged. In a tuple-based formulation, a construction can apply only to a sub-tuple, the remainder is left unchanged without an explicit identity constructor.

or automata) where the topology describes which outputs of one system are provided as an input to which other system. The concept of *abstract systems* as described in Section 1.1.4 captures such topologies.

We first formalize in Section 3.2.1 a system algebra that captures the notion of abstract systems. We continue in Section 3.2.2 to describe a restricted case, called the *cryptographic algebra*, whose specific axioms (basically, it restricts and concisely describes the topologies that are relevant for cryptographic protocols) are used to formulate and prove statements about protocols in a setting with multiple parties. Section 3.2.3 describes how the operations in the cryptographic algebra can be instantiated using the operations in any system algebra.

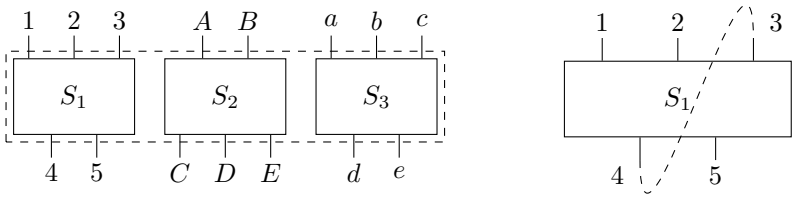
3.2.1 The Algebra of Abstract Systems

At the highest level of abstraction, following the hierarchy described by Maurer and Renner [MR11], systems are objects with interfaces by which they connect to (interfaces of) other systems. Each interface is labeled with an element of a given label set. The operations of the system algebra, i.e., which systems can be connected via which interfaces, then capture the topologies in which systems can be connected.

Mathematically, such a system algebra is a set $\mathcal{S}$ of systems together with a parallel composition operation that composes two or more systems into a single system,[2] and an operation for connecting two interfaces of a single system. Interfaces of different systems are connected by first composing them into a single system and then connecting the desired interfaces. The parallel composition is depicted in Figure 3.1a, where three systems S_1 (with interface labels $1, \ldots, 5$), S_2 (with labels $A, \ldots, E$) and S_3 (with labels $a, \ldots, e$) are composed. The interface labels of the systems are preserved during the composition; the resulting system thus has interfaces labeled by elements $1, \ldots, 5, A, \ldots, E, a, \ldots, e$.

For any pair of interface labels, there is an interface connecting operation which connects the two specified interfaces of a system. This is depicted in Figure 3.1b where the interfaces 3 and 4 of the system S_1 are connected. First connecting two interfaces of a system and then composing the resulting system in parallel with further systems is the same as first performing the parallel composition and then connecting the corresponding interfaces. If the order in which pairs of interfaces are connected is irrelevant, the algebra is called *connection-order invariant*. The described concept of system algebra is formalized in the following definition.

[2] This mapping need only be defined if the systems have distinct interface names.



(a) The system obtained by parallel composition of S_1, S_2, and S_3.

(b) The system obtained by connecting interfaces 3 and 4 of S_1.

Figure 3.1: Composition operations in the system algebra.

Definition 3.4. Let $\mathcal{J}$ be some label set. A *system algebra with interface set* $\mathcal{J}$ is a set $\mathcal{S}$ such that for each system $S \in \mathcal{S}$ there is an associated set $\mathcal{J}_S \subseteq \mathcal{J}$; we say that S *has interfaces* $\mathcal{J}_S$. A system algebra is *finitary* if for all $S \in \mathcal{S}$, the set $\mathcal{J}_S$ is finite. A system algebra has the following operations:

Parallel composition: A mapping $\| : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ that maps a pair of systems with disjoint interface sets to a single system such that with $S_1, S_2 \in \mathcal{S}$ with interface sets $\mathcal{J}_{S_1}$ and $\mathcal{J}_{S_2}$: $\mathcal{J}_{S_1 \| S_2} = \mathcal{J}_{S_1} \cup \mathcal{J}_{S_2}$.

Interface connecting: For each pair $j, j' \in \mathcal{J}$, an interface connecting operation $\gamma_{j:j'} : \mathcal{S} \to \mathcal{S}$ such that for each $S \in \mathcal{S}$ with

$$j, j' \in \mathcal{J}_S : \quad \mathcal{J}_{\gamma_{j:j'}(S)} \quad = \quad \mathcal{J}_S \setminus \{j, j'\},$$

that is, each interface can be connected to only a single other interface. The operation $\gamma_{j:j'}$ is the identity on systems S with $j, j' \notin \mathcal{J}_S$.

The parallel composition is associative and commutative, i.e.,

$$\forall S_1, S_2, S_3 \in \mathcal{S} : \ (S_1 \| S_2) \| S_3 \ = \ S_1 \| (S_2 \| S_3) \quad \text{and} \quad S_1 \| S_2 \ = \ S_2 \| S_1,$$

and the two types of mappings commute in the sense that

$$\forall S_1, S_2 \in \mathcal{S}, \ j, j' \in \mathcal{J}_{S_1} : \quad \gamma_{j:j'}(S_1) \| S_2 \quad = \quad \gamma_{j:j'}(S_1 \| S_2), \tag{3.1}$$

The algebra $\mathcal{S}$ is called *connection-order invariant* if additionally for all distinct $j_1, j_1', j_2, j_2' \in \mathcal{J}$, the operations $\gamma_{j_1:j_1'}$ and $\gamma_{j_2:j_2'}$ commute. $\diamond$

It is often helpful to *relabel* the interfaces in the description of complex scenarios where multiple systems are connected. Such relabeling operations are supposed to merely change the way in which the interfaces of the system are addressed, not to change the (behavior of the) system itself. Each relabeling operation can be described by a (possibly only partially defined) injective mapping $\rho : \mathcal{J} \to \mathcal{J}$ that maps the original interface names to the names after the relabeling is applied. By slight abuse of notation, we usually use the same symbol to describe the induced mapping $\rho : \mathcal{S} \to \mathcal{S}$.

Definition 3.5. A system algebra $\mathcal{S}$ with interface set $\mathcal{J}$ is called a *system algebra with relabeling operations* if each injective function $\rho : \mathcal{J} \to \mathcal{J}$ induces an operation $\mathcal{S} \to \mathcal{S}$ that maps a system S with interface set $\mathcal{J}_S$ to a system $\rho(S)$ with interface set $\rho(\mathcal{J}_S)$, such that

1. the transformation of mappings $\mathcal{J} \to \mathcal{J}$ to mappings $\mathcal{S} \to \mathcal{S}$ is a monoid homomorphism, i.e.

$$\forall S \in \mathcal{S}, \rho_1, \rho_2 : \mathcal{J} \to \mathcal{J} : \qquad \rho_2(\rho_1(S)) \quad = \quad (\rho_2 \circ \rho_1)(S);$$

2. for an operation $\gamma_{j:j'}$ and a relabeling mapping $\rho : \mathcal{J} \to \mathcal{J}$, first connecting the two specified interfaces and then relabeling the remaining ones is the same as first relabeling all interfaces and then connecting the relabeled interfaces, i.e.

$$\rho \circ \gamma_{j:j'} \quad = \quad \gamma_{\rho(j):\rho(j')} \circ \rho; \tag{3.2}$$

3. and for systems $S_1, S_2 \in \mathcal{S}$ and functions $\rho_1 : \mathcal{J}_{S_1} \to \mathcal{J}$ and $\rho_2 : \mathcal{J}_{S_2} \to \mathcal{J}$ such that $\mathcal{J}_{S_1} \cap \mathcal{J}_{S_2} = \emptyset$, $\rho_1(\mathcal{J}_{S_1}) \cap \rho_2(\mathcal{J}_{S_2}) = \emptyset$, and ρ_1 is the identity on $\mathcal{J}_{S_2}$ and ρ_2 is the identity on $\mathcal{J}_{S_1}$,

$$(\rho_1 \circ \rho_2)\,(S_1 \parallel S_2) \quad = \quad \rho_1(S_1) \parallel \rho_2(S_2), \tag{3.3}$$

   i.e., first composing two systems in parallel and then relabeling all interfaces is the same as first relabeling the interfaces of the systems individually and then composing them in parallel.

$\diamond$

The conditions correspond to the requirement that relabeling the interfaces of a system does not change the system itself, but only the way in which it is connected to other systems.

3.2.2 The Cryptographic Algebra

Cryptographic protocols are applied in scenarios where multiple parties interact. We explicitly formalize the *resources* that are available to the parties; in a setting with multiple parties, resources are potentially accessible to all involved parties. Specific resources which are available only locally to one party can be considered as a special case. In terms of abstract systems, this means that a resource is a system that provides one interface to each of the parties. We will usually denote resources either by symbols such as $\bullet\!\!-\!\!\rightarrow$ or by upper case letters.

An algorithm or method that is applied by a party and makes use of the resource can be seen as a two-interface system that connects with its *inside* interface to the resource and provides its *outside* interface to the party. Such a system is called a *converter* and, at a lower abstraction layer, the inside interface specifies the way in which the converter uses the resource, the outside interface specifies in which way it can be used by the party or by higher-level protocols. The term "converter" reflects the fact that it "converts" the interface provided by the resource into a different interface. The composition of a resource and a converter is again a resource that provides one interface to each party. Converters are denoted either by small Greek letters or by special identifiers such as enc or dec.

The described structure can be captured algebraically by considering a set Φ of resources and a set Σ of converters together with the described operation. We refer to this algebraic structure as *cryptographic algebra*; the concept was introduced by Maurer and Renner [MR11]. The operation of attaching the inside interface of the converter $\alpha \in \Sigma$ to interface I of the resource R is written as $\alpha^I R$.

Definition 3.6 (Cryptographic algebra)**.** A *cryptographic algebra with interface set* $\mathcal{I}$ is a pair (Φ, Σ) of sets of resources Φ and converters Σ, with a (potentially partially defined) converter application operation. For each $I \in \mathcal{I}$, this is an operation

$$\cdot^I \cdot : \quad \Sigma \times \Phi \to \Phi, \quad (\alpha, R) \mapsto \alpha^I R.$$

The algebra is *composition-order invariant* if

$$\forall \alpha, \beta \in \Sigma,\ R \in \Phi,\ I \neq J \in \mathcal{I} : \qquad \alpha^I \beta^J R \quad = \quad \beta^J \alpha^I R.$$

◇

Composition-order invariance ensures that drawing a diagram such as the one in Figure 3.2 makes sense: The figure does not encode the information whether the converter α or the converter β is attached to the resource first,

but composition-order invariance guarantees that both orders result in the same resource.

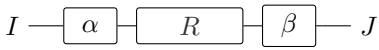


Figure 3.2: A resource R with converters α and β connected at interfaces I and J, respectively. This structure is denoted as $\alpha^I \beta^J R$.

Serial composition of converters. For a resource $R \in \Phi$, a converter $\alpha \in \Sigma$, and an interface $I \in \mathcal{I}$, the term $\alpha^I R$ describes again a resource. If a further converter $\beta \in \Sigma$ is attached to the same interface $I \in \mathcal{I}$, the resulting object $\beta^I \left(\alpha^I R\right)$ is again a resource. More generally, one can consider sequences of converters, with the understanding that a sequence of converters is attached to the resource in some pre-described order. We write the sequence in the above expression as $\beta \circ \alpha$ and define

$$(\beta \circ \alpha)^I R \quad := \quad \beta^I \left(\alpha^I R\right). \tag{3.4}$$

The operation "$\circ$" is associative by definition. In this sense, it is sound to consider sequences of converters, i.e., elements of Σ^* as "a converter," since composition-order invariance guarantees that converter applications at different interfaces commute. It is not generally necessary to require that a sequence of converters is again a single converter; however, for many natural instantiations, like the one we describe in Section 3.2.3, such an operation can actually be defined.

Parallel composition: Tuples of resources and converters. In most settings, parties can have access to multiple resources and run several protocols simultaneously. This is formalized by parallel composition operations on the sets of resources and converters, and for resources it means that, for instance, a communication channel and a cryptographic key are available. For converters, the parallel composition corresponds to applying the converters to the available resources individually, such as an encoding to the communication channel and an expansion to the key.

If multiple resources are available, a party must be able to address the particular resource to which it wants to apply a converter. Hence, the availability of multiple resources corresponds to the availability of a *tuple* of resources, where the labels of the tuple allow to address the resource to which a converter shall be attached. In a cryptographic algebra, this means that for a

set $L \subseteq \Lambda$ of labels and resources $R_\lambda \in \Phi$ for each $\lambda \in L$, the tuple $\langle R_\lambda \rangle_{\lambda \in L}$ is again a resource; we write this as $\langle R_\lambda \rangle_{\lambda \in L} \in \Phi$. Each resource R_λ in the tuple is identified by its label $\lambda \in L$.

In the same spirit, for a set $L' \subseteq \Lambda$ of labels and converters $\alpha_\lambda \in \Sigma$ for each $\lambda \in L'$, one can consider a tuple $\langle \alpha_\lambda \rangle_{\lambda \in L'}$ of converters. As for the resources, this means that $\langle \alpha_\lambda \rangle_{\lambda \in L'} \in \Sigma$. The capability of a party to address a specific resource corresponds to applying a tuple of converters where the resources and converters with the same label are composed; this is formalized by requiring a corresponding condition in the definition of a cryptographic algebra with parallel composition, as formalized in equation (3.5).

Definition 3.7. For a label set Λ, a cryptographic algebra (Φ, Σ) is a *cryptographic algebra with Λ-parallel composition* if the sets Φ and Σ are closed under taking finite tuples and for $L' \subseteq L \subseteq \Lambda$ such that $\langle \alpha_\lambda \rangle_{\lambda \in L'}$ is a tuple of converters and $\langle R_\lambda \rangle_{\lambda \in L}$ is a tuple of resources, the equation

$$\langle \alpha_\lambda \rangle_{\lambda \in L'}{}^{I} \langle R_\lambda \rangle_{\lambda \in L} \quad = \quad \langle R_\lambda \rangle_{\lambda \in L \setminus L'} \cup \left\langle \alpha_\lambda{}^{I} R_\lambda \right\rangle_{\lambda \in L'}, \tag{3.5}$$

holds. ◇

We introduce special notation for denoting the situation where multiple copies of the same resource are available. For a resource $R \in \Phi$, we denote by $\langle R \rangle$ the unbounded sequence of copies of R, where the copies are labeled by $1, 2, \ldots \in \mathbb{N}$, that is,

$$\langle R \rangle \quad := \quad (R, R, \ldots).$$

We write $\langle R \rangle_{[n]}$ if n copies of the resource R are available with labels $1, 2, \ldots \in [n]$, that is,

$$\langle R \rangle_{[n]} \quad := \quad (R, \ldots, R),$$

where the sequence has n elements.

Viewing parallel composition as an operation. Based on the parallel composition, we define an operation on resources that is described by "adding" one ore more specified resources in parallel. For the simple case where we only "add" a single resource S, this operation can be written as $(\cdot, S) : \Phi \to \Phi,\ R \mapsto (R, S)$ and is understood as taking S in parallel to a resource specified on the right.[3] More generally, for $n \in \mathbb{N}$, resources $R_1, \ldots, R_n \in \Phi$, and $i \leq n$, the term $(R_1, \ldots, R_{i-1}, \cdot, R_{i+1}, \ldots, R_n)$ is defined to denote the operation of taking the resources $R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n$ in parallel to the

[3]For notational simplicity, we restrict ourselves to tuples which are indexed by integers; the extension to generic label sets is straightforward.

resource specified on the right. In particular, for the setting described above, this means

$$(R_1, \ldots, R_{i-1}, \cdot, R_{i+1}, \ldots, R_n)\, R_i \quad := \quad (R_1, \ldots, R_n)\,. \tag{3.6}$$

The following lemma formalizes in which sense the operation of taking resources in parallel commutes with the converter application operation, it is stated with respect to the operation defined on a single resource $(\cdot, S)$, the analogous statements hold with respect to lists of arbitrary length, arbitrary "empty positions" in the list, and more generally tuples with arbitrary labels.

Lemma 3.8. *Let (Φ, Σ) be a cryptographic algebra with Λ-parallel composition. For $\alpha \in \Sigma$, $I \in \mathcal{I}$, and $R, S \in \Phi$:*

$$(\cdot, S)\, \alpha^I R = \langle \alpha \rangle_1{}^I\, (\cdot, S)\, R,$$

where the notation $\langle \alpha \rangle_1$ denotes a tuple that has a single element α with label 1, as a set $\langle \alpha \rangle_1 = \{(1, \alpha)\}$.

Proof. The property follows directly from the composition of the operation $(\cdot, S)$ and Definition 3.7. In more detail,

$$(\cdot, S)\, \alpha^I R \quad = \quad \left(\alpha^I R, S\right) \quad = \quad \langle \alpha \rangle_1{}^I\, (R, S) \quad = \quad \langle \alpha \rangle_1{}^I\, (\cdot, S)\, R,$$

where the first and third equality use equation (3.6) and the second equality uses equation (3.5). □

3.2.3 Instantiating the Operations of the Cryptographic Algebra

The cryptographic algebra can be seen as describing a specific type of topology on abstract systems. Resources and converters are abstract systems, and the composition operation that connects a converter system to a resource system is defined based on their connection as abstract systems. Recall that a cryptographic algebra with interface set $\mathcal{I}$ consists of a set Φ of resource systems and a set Σ of converter systems, together with a converter application operation $\cdot^I \cdot : \Sigma \times \Phi \to \Phi$ for each interface $I \in \mathcal{I}$.

Resources and converters. Let $\mathcal{I}$ and $\mathcal{L}$ be label sets and $\mathsf{In}, \mathsf{Out} \notin \mathcal{I}$ be two constants. Let $\mathcal{S}$ be a finitary abstract system algebra with interface set $\mathcal{J} = (\mathcal{I} \cup \{\mathsf{In}, \mathsf{Out}\}) \times \mathcal{L}$. We define the set of resource systems to be the set of all systems with interface labels in the set $\mathcal{I} \times \mathcal{L}$, that is,

$$\Phi \quad := \quad \{S \in \mathcal{S} : \mathcal{J}_S \subseteq \mathcal{I} \times \mathcal{L}\}\,,$$

and the set of converter systems to be the set of all systems with interface labels in the set $\{\mathsf{In}, \mathsf{Out}\} \times \mathcal{L}$, that is,

$$\Sigma \quad := \quad \{S \in \mathcal{S} : \mathcal{J}_S \subseteq \{\mathsf{In}, \mathsf{Out}\} \times \mathcal{L}\}\,.$$

To understand resources as systems with interfaces in $\mathcal{I}$ (instead of $\mathcal{I} \times \mathcal{L}$) and converters as systems with interfaces in $\{\mathsf{In}, \mathsf{Out}\}$ (instead of $\{\mathsf{In}, \mathsf{Out}\} \times \mathcal{L}$), we consider each interface in $\mathcal{I}$ or $\{\mathsf{In}, \mathsf{Out}\}$ as consisting of sub-interfaces which are labeled by *sub-interface labels* in $\mathcal{L}$ relative to the interface. Two interfaces (such as $I \in \mathcal{I}$ and In) are then connected by connecting all sub-interfaces with the same "local" label $\lambda \in \mathcal{L}$, as described in the next paragraph. More generally, for interface $I \in \mathcal{I} \cup \{\mathsf{In}, \mathsf{Out}\}$ and each subset $L \subseteq \mathcal{L}$, we can consider the sub-interface described by all labels in the set $\{I\} \times L$.[4]

This interface structure is depicted in Figure 3.3 which shows two systems α and R, where α has interface labels in $\{\mathsf{In}, \mathsf{Out}\} \times \mathbb{N}$ and R has interface labels in $\{A, B, E\} \times \mathbb{N}$. Hence, the system α is considered a converter system, whereas R is considered a resource system. The dotted ellipses indicate that all interfaces whose labels coincide in the first component are considered as being "sub-interfaces" of the interface which is labeled by that first component.

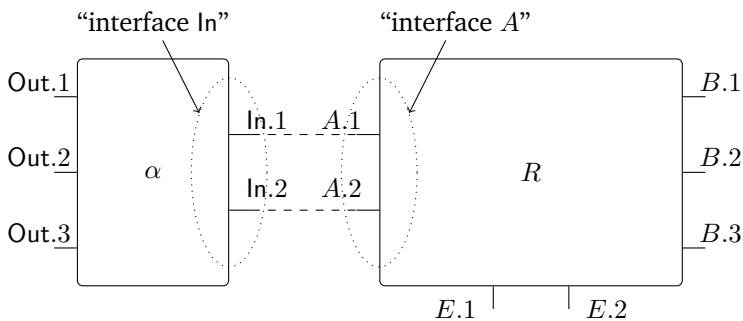


Figure 3.3: A converter system α and a resource system R, where the In-interface of α is connected to the A-interface of R.

Composition operation. The composition operation for a converter system $\alpha \in \Sigma$, a resource system $R \in \Phi$, and an interface $I \in \mathcal{I}$ is supposed to connect the I-interface of the resource system R to the In-interface of the

[4]For an interface $I \in \mathcal{I}$ and a "local" label $\boldsymbol{\lambda}$, we usually write the pair $(I, \boldsymbol{\lambda})$ as the concatenation $I.\boldsymbol{\lambda}$.

converter system α. We only define the composition operation if the set of sub-interfaces is the same for the two connected interfaces; that is, if there is a finite set $L \subseteq \mathcal{L}$ such that $L = \{\lambda \in \mathcal{L} \mid I.\lambda \in \mathcal{J}_R\} = \{\lambda \in \mathcal{L} \mid \mathsf{In}.\lambda \in \mathcal{J}_\alpha\}$.

The composition is then obtained by taking the parallel composition of R and α and connecting the interfaces appropriately and modifying the interface labels by the mapping $\rho_I : \mathsf{Out}.\lambda \mapsto I.\lambda$, which relabels the outside interface of the converter system to become the I-interface of the composed system. Altogether, the composition is defined as

$$\alpha^I R \coloneqq \left(\rho_I \circ \prod\nolimits_{\lambda \in L} \gamma_{\mathsf{In}.\lambda : I.\lambda}\right)(\alpha \parallel R)\,. \tag{3.7}$$

The connection of a converter system α and a resource system R—without the relabeling—is depicted in Figure 3.3. The system $\alpha^A R$ is obtained by relabeling the interfaces $\mathsf{Out}.i$ to $A.i$, for $i \in \{1, 2, 3\}$.

Tuples. To formally define the parallel composition, we consider the special case where the set $\mathcal{L}$ is defined as $\mathcal{L} \coloneqq \mathbf{\Lambda} = \Lambda^*$ for some label set Λ. As the parallel composition of resource systems (or converter systems) is defined via tuples with explicit labels, these labels can be used to access the specific sub-interface of a particular resource system (or converter system) in a parallel composition. More concretely, if a resource system R appears in a tuple with label λ, then all sub-interfaces of this resource system are relabeled by prepending the label λ to the prior label of the sub-interface. An exemplary setting with a tuple of 2 resource systems is depicted in Figure 3.4.

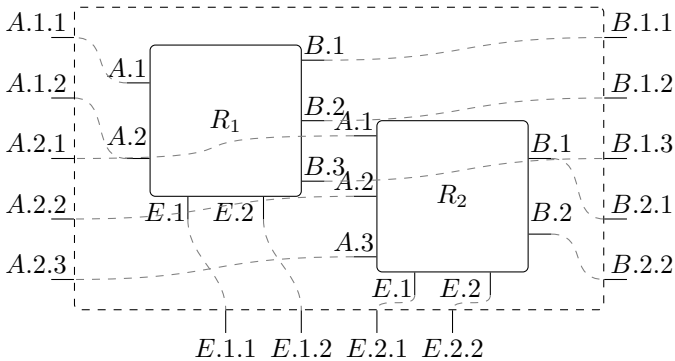


Figure 3.4: Interface relabeling for the tuple $\{(1, R_1), (2, R_2)\}$ of resource systems.

In the cryptographic algebra, for a tuple of resource systems $\langle R_\lambda \rangle_{\lambda \in L}$ with labels in $L \subseteq \Lambda$, the term $\left(\langle R_\lambda \rangle_{\lambda \in L}\right)$ is defined as first relabeling each R_λ via

$I.\boldsymbol{\lambda} \mapsto I.\lambda.\boldsymbol{\lambda}$ and then taking the resulting systems in parallel. The tuple of converter systems is defined analogously.

The following lemma shows that the resulting algebra is indeed a composition-order independent cryptographic algebra with Λ-parallel composition.

Lemma 3.9. *Let $\mathcal{S}$ be a connection-order invariant finitary system algebra with interface set $\mathcal{J} = (\mathcal{I} \cup \{\mathsf{In}, \mathsf{Out}\}) \times \boldsymbol{\Lambda}$, and let Φ and Σ be as above. Then, the algebra (Φ, Σ) is a composition-order invariant cryptographic algebra with interface set $\mathcal{I}$ and Λ-parallel composition.*

Proof. The composition of a resource system $R \in \Phi$ and a converter system $\alpha \in \Sigma$ is indeed a resource system; as a system it has interfaces in $\mathcal{I} \times \boldsymbol{\Lambda}$ because the In-interface of the converter system is connected to the I-interface of the resource system, and the Out-interface is relabeled.

We still have to show that the algebra is composition-order invariant, which means that the terms $\alpha^I \beta^J R$ and $\beta^J \alpha^I R$ describe the same system. To verify this, we define relabeling mappings that change the names of the interfaces of α and β such that the name collisions are prevented; denote by $\hat{\rho}_I$ and $\hat{\rho}_J$ be two injective mappings such that $\operatorname{im} \hat{\rho}_I \cap \operatorname{im} \hat{\rho}_J = \emptyset$ and the assigned names also do not collide with those of R (this can be defined since $\boldsymbol{\Lambda} = \Lambda^*$). Then, we obtain

$$\alpha^I \beta^J R = \left[\rho_I \circ \prod_{\lambda \in L} \gamma_{\mathsf{In}.\lambda : I.\lambda}\right] \left(\alpha \parallel \left[\rho_J \circ \prod_{\lambda \in L} \gamma_{\mathsf{In}.\lambda : J.\lambda}\right] (\beta \parallel R)\right) \tag{3.8}$$

$$= \left[\rho_I \circ \prod_{\lambda \in L} \gamma_{\mathsf{In}.\lambda : I.\lambda}\right] \left(\hat{\rho}_I^{-1} \hat{\rho}_I(\alpha) \parallel \left[\rho_J \circ \prod_{\lambda \in L} \gamma_{\mathsf{In}.\lambda : J.\lambda}\right] \left(\hat{\rho}_J^{-1} \hat{\rho}_J(\beta) \parallel R\right)\right) \tag{3.9}$$

$$= \left[\rho_I \circ \prod_{\lambda \in L} \gamma_{\mathsf{In}.\lambda : I.\lambda} \circ \hat{\rho}_I^{-1}\right] \left(\hat{\rho}_I(\alpha) \parallel \left[\rho_J \circ \prod_{\lambda \in L} \gamma_{\mathsf{In}.\lambda : J.\lambda} \circ \hat{\rho}_J^{-1}\right] (\hat{\rho}_J(\beta) \parallel R)\right) \tag{3.10}$$

$$= \left[\rho_I \hat{\rho}_I^{-1} \circ \prod_{\lambda \in L} \gamma_{\hat{\rho}_I(\mathsf{In}.\lambda) : I.\lambda}\right] \left(\hat{\rho}_I(\alpha) \parallel \left[\rho_J \hat{\rho}_J^{-1} \circ \prod_{\lambda \in L} \gamma_{\hat{\rho}_J(\mathsf{In}.\lambda) : J.\lambda}\right] (\hat{\rho}_J(\beta) \parallel R)\right) \tag{3.11}$$

$$= \left[\rho_I \hat{\rho}_I^{-1} \circ \prod_{\lambda \in L} \gamma_{\hat{\rho}_I(\mathsf{In}.\lambda):I.\lambda} \circ \rho_J \hat{\rho}_J^{-1} \circ \prod_{\lambda \in L} \gamma_{\hat{\rho}_J(\mathsf{In}.\lambda):J.\lambda} \right] (\hat{\rho}_I(\alpha) \parallel (\hat{\rho}_J(\beta) \parallel R)) \tag{3.12}$$

$$= \left[\rho_J \hat{\rho}_J^{-1} \circ \prod_{\lambda \in L} \gamma_{\hat{\rho}_J(\mathsf{In}.\lambda):J.\lambda} \circ \rho_I \hat{\rho}_I^{-1} \circ \prod_{\lambda \in L} \gamma_{\hat{\rho}_I(\mathsf{In}.\lambda):I.\lambda} \right] (\hat{\rho}_J(\beta) \parallel (\hat{\rho}_I(\alpha) \parallel R)) \tag{3.13}$$

$$= \cdots = \beta^J \alpha^I R,$$

where the steps indicated by "..." are exactly the same as before, in opposite order. Overall, we have used in (3.8) the definition from equation (3.7), in (3.9) that $\hat{\rho}_I$ and $\hat{\rho}_J$ are injective, in (3.10) that the ranges of $\hat{\rho}_I$ and $\hat{\rho}_J$ do not collide mutually or with the labels of R and equation (3.3), in (3.11) equation (3.2), in (3.12) equations (3.3) and (3.1), and in (3.13) the commutativity of the parallel composition, the connection-order invariance of the system algebra, and equation (3.11).

The condition specified in Definition 3.7 follows from the fact that the tuples of resource systems and converter systems are defined analogously, that is, since the applied relabeling mappings are consistent, the statement follows from equation (3.2). □

3.3 The Construction Notion for Settings with a Single Attacker

The goal of protocols for secure communication is to allow two or more honest parties to communicate securely, even in presence of an attacker that attempts to eavesdrop or to disturb the communication. In this section, we describe a (composable) construction notion in the sense of Section 3.1 for the described setting. In Section 3.3.2, we describe a set of components Ω, a set of constructors Γ with a composition operation. We define what it means for a triple $(R, a, S) \in \Omega \times \Gamma \times \Omega$ to be valid according to the construction notion, and show that the described construction notion is composable in the sense of Definition 3.3. In Section 3.3.3, we show constructive statements can be parametrized to capture that certain parameters such as the key length or the number of transmitted messages are reflected in the security bound.

3.3.1 A Distinction Problem on Resources

The construction notion we describe in this section is, as explained in Section 1.1.3, based on the "real-world/ideal-world" paradigm. Both the "real world" and the "ideal world" are described in terms of resources and converters, which by Section 3.2.2 means that they are formally resources.

For protocols that do not achieve perfect security, for example because they are based on computational assumptions, the two considered "worlds" will not be equivalent. Rather, we will consider a distinction problem between the two resources, following the formalization in Section 2.2. Both resources are considered as systems to which a distinguisher (also a system) can be attached, such that the composition of the resource and the distinguisher determines a bit. We denote this set of distinguishers for abstract resources as $\mathcal{D}$, and a performance measure in this setting is formally a mapping $\mathcal{D} \to [0, 1]$ that assigns to each distinguisher the advantage that it achieves.

The explicit appearance of the performance measures is necessary in a "concrete," i.e. non-asymptotic, formulation of security. Since the run-time of systems and the advantages are not defined asymptotically, one cannot "absorb efficient systems" into the distinguisher, and the distinguishing advantage cannot be required to be "negligible." This becomes apparent in the composition theorem, where both converters and the error terms accumulate if multiple constructions are composed.

3.3.2 The Notion of Construction

From a constructive perspective, the goal of a cryptographic protocol is to construct a desired resource from one or more assumed resources. We focus on the case where two (or more) honest parties and potentially one attacker are present, and hence consider resources which provide one interface to each honest party and one interface to the attacker. A protocol in this setting is a tuple of converters; there is (at most[5]) one converter for each honest party.

Definition 3.10. For a set of interface labels $\mathcal{I}$ with $E \in \mathcal{I}$, a *protocol for $\mathcal{I}$-resources with attacker E* is a tuple $\boldsymbol{\pi} \in \Pi$ with

$$\Pi \coloneqq \Sigma^{(\mathcal{I}')} \qquad \text{with} \qquad \mathcal{I}' = \mathcal{I} \setminus \{E\},$$

i.e., a protocol $\boldsymbol{\pi}$ is a tuple $\boldsymbol{\pi} = \langle \pi_I \rangle_{I \in \mathcal{I}''}$ with $\mathcal{I}'' \subseteq \mathcal{I} \setminus \{E\}$. ◇

[5] A protocol may also specify that a certain party does not apply a converter but uses the resources unmodified.

The application of a protocol $\boldsymbol{\pi}$ to a resource R is then defined as attaching the converters to the respective honest interfaces:

$$\boldsymbol{\pi} R \coloneqq \pi_{I_1}{}^{I_1} \ldots \pi_{I_n}{}^{I_n} R \qquad \text{with} \qquad \mathcal{I}'' = \{I_1, \ldots, I_n\} = \operatorname{supp} \boldsymbol{\pi}.$$

The term $\boldsymbol{\pi} R$ is well-defined for composition-order invariant cryptographic algebras because the order in which the converters are attached to the resource is irrelevant. We usually write the protocols as a pair or sequence if the labels to which the elements of the sequence correspond are clear from the context.

Analogously to the serial application of converters to a resource in equation (3.4), we can define the serial application of protocols to a resource. By composition-order invariance, this simply means that at each interface of the resource one attaches the sequence of converters that one obtains by composing the protocols by composing the converters that comprise the protocols per interface.

Availability and security. A protocol must satisfy two requirements: First, the protocol must construct the desired resource in a setting where no attacker is present. This condition is referred to as *availability* or *correctness* condition and excludes trivial protocols that do not even achieve the desired functionality if the protocol is not attacked. Second, the condition must construct the desired resource in a setting where an attacker attempts to, for instance, eavesdrop or to disturb the communication. This condition is referred to as *security* condition.

The two conditions can be viewed as a special case of the more general definition of construction in the abstract cryptography framework [MR11] for the case where only one party is potentially dishonest, and the security condition be viewed as an instance of the "real-world/ideal-world" paradigm. The "real world" is described by the protocol $\boldsymbol{\pi}$ and the assumed resource R; it is denoted as $\boldsymbol{\pi} R$. The honest parties access the outside interfaces of the converters, the attacker directly accesses the resource. While the converters are expected to mimic the interfaces of the constructed resource S, the E-interfaces of $\boldsymbol{\pi} R$ and S will in general not be similar, and hence the distinction problem between the two resources $\boldsymbol{\pi} R$ and S is easy to solve. This is compensated by means of a *simulator*, which is a converter (in the cryptographic algebra) that is attached to the E-interface of S. The security condition requires the systems $\boldsymbol{\pi} R$ and $\sigma^E S$ to be similar, and the underlying intuition is that an attacker accessing the E-interface of $\boldsymbol{\pi} R$ can launch the same attack in a setting with the resource S by attaching the converter σ at its interface of the resource and then launching the attack.

An axiomatic derivation of the two described conditions and the simulation paradigm is described by Maurer and Renner [MR11, Sections 5, C,

and E]. An exemplary description of the security condition for the one-time pad can be found in Section 1.1.3 and by Maurer [Mau11].

Instantiating the components. The component set Ω is defined in terms of resources in a cryptographic algebra. Since the construction notion is supposed to capture both an *availability* and a *security* condition, one needs to specify the behavior of a resource both in the case where no attacker is present, and in the case where an attacker is present and potentially attempts to attack the protocol. There are several ways in which the two guarantees of a resource can be specified for these two conditions:

Guaranteed converter: For each resource $R \in \Phi$, one specifies a converter $\perp_R$ that specifies the assumed "behavior" of the attacker when it is absent. (The behavior of the resource in that case is ${\perp_R}^E R$.) This models the fact that the attacker can always choose to not launch an attack and the formalization closely follows the more abstract definition of Maurer and Renner [MR11]. A resource is then specified by the pair $(R, \perp_R)$ of a resource system and a converter system.

Pairs of systems: A resource can be described as a pair of resource systems, one for each condition. This formally means that to each resource $R \in \Phi$ we assign a second resource $R_\perp$ that formalizes the behavior of R in the setting where no attacker is present, and corresponds to ${\perp_R}^E R$ above. If R is an $\mathcal{I}$-resource, then $R_\perp$ is an $(\mathcal{I} \setminus \{E\})$-resource, and for this description to be meaningful in the sense of [MR11], one actually has to show that such a converter exists. Since this method results in simple and easy to grasp descriptions of resources, we follow this approach in the remainder of the thesis.

Resources with two modes: A resource can be described as operating in two different modes which can be selected via a special input, which was referred to as "cheating bit" in several papers, e.g. [MRT12, CMT13a]. There is a special converter $\perp$ which sets the "cheating bit" to 0, whereas the attacker can choose any of the two modes. While this formalization allows to specify the resource as a single object, the formulation is sometimes misleading because the "cheating bit" should not be an (adaptively selectable) input. Also, the description of the resources tend to be more complex.

For the remainder of the thesis, we define a resource as a pair of resource systems $(R, R_\perp)$; more formally, we set $\Omega = \Phi \times \Phi$.[6] This choice only affects

[6] In the context of construction statements, we will usually refer to this pair of resource systems also as a "resource."

the presentation; all formulations result in equivalent conditions. While a resource is formally a pair of two resource systems $(R, R_\perp)$, we usually only refer to the resource by R and consider $R_\perp$ as being implicitly understood.

Instantiating the constructors. A constructive statement consist of the assumed resource R, the constructed resource S, the (sequence of) protocol(s) supposed to achieve the construction, and the information "how well" the protocol(s) achieve this construction. This information is captured in the simulator (which may be a sequence of converters) that appears in the security condition, and in bounds for the distinction advantage between the "real world" and the "ideal world." Consequently, a *constructor* contains the sequence of protocols, the simulator, and the performance measures that bound the distinguishing advantages in the availability and the security conditions.

Definition 3.11. For a set of interface labels $\mathcal{I}$ with $E \in \mathcal{I}$, a *constructor for $\mathcal{I}$-resources with attacker E* is a triple $a \in \Gamma$ with

$$\Gamma \quad \coloneqq \quad \Pi^* \times \Sigma^* \times (\mathcal{D} \to [0,1])^2 \,,$$

where $\mathcal{D}$ denotes the set of all distinguishers in the sense of Section 2.2. The composition operation $\circ$ on the set Γ is defined as, for $a = (\boldsymbol{\pi}, \sigma_{\boldsymbol{\pi}}, \varepsilon_{\boldsymbol{\pi}}) \in \Gamma$ and $b = (\boldsymbol{\psi}, \sigma_{\boldsymbol{\psi}}, \varepsilon_{\boldsymbol{\psi}}) \in \Gamma$,

$$b \circ a \quad \coloneqq \quad (\boldsymbol{\psi} \circ \boldsymbol{\pi}, \sigma_{\boldsymbol{\pi}} \circ \sigma_{\boldsymbol{\psi}}, \varepsilon) \,,$$

with $\varepsilon = \left(\varepsilon^1, \varepsilon^2\right)$ for $\varepsilon^1 = \varepsilon^1_{\boldsymbol{\pi}} \circ (\cdot\boldsymbol{\psi}) + \varepsilon^1_{\boldsymbol{\psi}}$ and $\varepsilon^2 = \varepsilon^2_{\boldsymbol{\pi}} \circ (\cdot\boldsymbol{\psi}) + \varepsilon^2_{\boldsymbol{\psi}} \circ (\cdot{\sigma_{\boldsymbol{\pi}}}^E)$. $\diamond$

The reason for the exact definition of the composition operation may not be clear immediately; in fact, the definition of the obtained performance measure ε is chosen such that the construction can be shown to be composable in Theorem 3.13. It makes explicit that in the proof of the composition theorem, the protocol $\boldsymbol{\psi}$ and the simulator $\sigma_{\boldsymbol{\pi}}$ have to be "absorbed" into the distinguisher and hence affect the performance measure.

Instantiating the construction notion. The construction notion is formally a subset of $\Omega \times \Gamma \times \Omega$ (it can be considered as relation described by triples) that specifies which constructions are considered to be "valid." We define the construction notion, for the setting with parties labeled by elements in $\mathcal{I}$ and an attacker E, as follows. Let $a = (\boldsymbol{\pi}, \sigma, \varepsilon)$ with $\varepsilon = \left(\varepsilon^1, \varepsilon^2\right)$ be a constructor and let $(R, R_\perp)$ and $(S, S_\perp)$ be resources. The availability condition then states that the distinguishing advantage between protocol $\boldsymbol{\pi}$ applied to $R_\perp$ and the constructed resource $S_\perp$ must be bounded by ε^1, and the security

condition states that the distinguishing advantage between the protocol $\boldsymbol{\pi}$ applied to R and the simulator σ applied to S is bounded by ε^2.

Definition 3.12. A protocol $\boldsymbol{\pi} \in \Pi^*$ *constructs the resource* S *from the assumed resource* R*, with respect to simulator* $\sigma \in \Sigma^*$ *and within* $\varepsilon = (\varepsilon_1, \varepsilon_2)$, denoted $R \overset{\boldsymbol{\pi},\sigma,\varepsilon}{\Longrightarrow} S$, if

$$[\![(\boldsymbol{\pi} R_\perp \mid S_\perp)]\!] \quad \leq \quad \varepsilon_1 \qquad \text{and} \qquad [\![(\boldsymbol{\pi} R \mid \sigma^E S)]\!] \quad \leq \quad \varepsilon_2,$$

where the equation holds with respect to the set $\mathcal{D}$ of all distinguishers (in the sense of Section 2.2). $\diamond$

The application of sequences of protocols and converters to a resource is defined as in equation (3.4).

The composition theorem. An important property of Definition 3.12 is its composability. Theorem 3.13 shows that the construction notion is indeed composable, that is, compatible with the composition on the set of constructors. The main technical argument that is required is the triangle inequality for the distinguishing advantage.

Theorem 3.13 (Composition theorem)**.** *Let* R*,* S*, and* T *be resources, and let* $\boldsymbol{\pi}$ *and* $\boldsymbol{\psi}$ *be protocols,* $\sigma_{\boldsymbol{\pi}}$ *and* $\sigma_{\boldsymbol{\psi}}$ *be simulators, and* $\varepsilon_{\boldsymbol{\pi}}$ *and* $\varepsilon_{\boldsymbol{\psi}}$ *pairs of performance measures such that*

$$R \overset{\boldsymbol{\pi},\sigma_{\boldsymbol{\pi}},\varepsilon_{\boldsymbol{\pi}}}{\Longrightarrow} S \qquad \textit{and} \qquad S \overset{\boldsymbol{\psi},\sigma_{\boldsymbol{\psi}},\varepsilon_{\boldsymbol{\psi}}}{\Longrightarrow} T.$$

Then

$$R \overset{\boldsymbol{\psi}\circ\boldsymbol{\pi},\sigma_{\boldsymbol{\pi}}\circ\sigma_{\boldsymbol{\psi}},\varepsilon}{\Longrightarrow} T,$$

where $\varepsilon^1 = \varepsilon^1_{\boldsymbol{\pi}} \circ (\cdot\boldsymbol{\psi}) + \varepsilon^1_{\boldsymbol{\psi}}$ *and* $\varepsilon^2 = \varepsilon^2_{\boldsymbol{\pi}} \circ (\cdot\boldsymbol{\psi}) + \varepsilon^2_{\boldsymbol{\psi}} \circ (\cdot{\sigma_{\boldsymbol{\pi}}}^E)$.

Proof. By definition, $[\![(\boldsymbol{\pi} R_\perp \mid S_\perp)]\!] \leq \varepsilon^1_{\boldsymbol{\pi}}$ and $[\![(\boldsymbol{\pi} R \mid {\sigma_{\boldsymbol{\pi}}}^E S)]\!] \leq \varepsilon^2_{\boldsymbol{\pi}}$ by the construction statement for $\boldsymbol{\pi}$ and $[\![(\boldsymbol{\pi} S_\perp \mid T_\perp)]\!] \leq \varepsilon^1_{\boldsymbol{\psi}}$ and $[\![(\boldsymbol{\pi} S \mid {\sigma_{\boldsymbol{\psi}}}^E T)]\!] \leq \varepsilon^2_{\boldsymbol{\psi}}$ by the construction statement for $\boldsymbol{\psi}$. Hence, we have that

$$[\![(\boldsymbol{\psi}\boldsymbol{\pi} R_\perp \mid T_\perp)]\!] \;\leq\; [\![(\boldsymbol{\psi}\boldsymbol{\pi} R_\perp \mid \boldsymbol{\psi} S_\perp)]\!] + [\![(\boldsymbol{\psi} S_\perp \mid T_\perp)]\!] \;\leq\; \varepsilon^1_{\boldsymbol{\pi}} \circ (\cdot\boldsymbol{\psi}) + \varepsilon^1_{\boldsymbol{\psi}},$$

by Lemma 2.6 and

$$\begin{aligned} &\Big[\!\Big[\Big(\boldsymbol{\psi}\boldsymbol{\pi} R \mid (\sigma_{\boldsymbol{\pi}} \circ \sigma_{\boldsymbol{\psi}})^E T\Big)\Big]\!\Big] \\ &\qquad \leq \quad [\![(\boldsymbol{\psi}\boldsymbol{\pi} R \mid \boldsymbol{\psi}{\sigma_{\boldsymbol{\pi}}}^E S)]\!] + \Big[\!\Big[\Big({\sigma_{\boldsymbol{\pi}}}^E \boldsymbol{\psi} S \mid (\sigma_{\boldsymbol{\pi}} \circ \sigma_{\boldsymbol{\psi}})^E T\Big)\Big]\!\Big] \end{aligned}$$

$$
\begin{aligned}
= \quad & [\![(\boldsymbol{\psi}\boldsymbol{\pi}R \mid \boldsymbol{\psi}\sigma_{\boldsymbol{\pi}}{}^{E}S)]\!] + [\![(\sigma_{\boldsymbol{\pi}}{}^{E}\boldsymbol{\psi}S \mid \sigma_{\boldsymbol{\pi}}{}^{E}(\sigma_{\boldsymbol{\psi}}{}^{E}T))]\!] \\
= \quad & \varepsilon^2_{\boldsymbol{\pi}} \circ (\cdot\boldsymbol{\psi}) + \varepsilon^2_{\boldsymbol{\psi}} \circ (\cdot\sigma_{\boldsymbol{\pi}}{}^{E}),
\end{aligned}
$$

by the definition of $\sigma_{\boldsymbol{\pi}} \circ \sigma_{\boldsymbol{\psi}}$ and Lemma 2.6, which verifies the conditions. □

The construction notion is *not* context insensitive in the sense of [MR11]. That means that if an additional resource T is available in parallel to the construction of a resource S from a resource R, then the construction will not hold with respect to the same bounds on the distinguishing advantage. The reason is that the construction notion is based on the notion of reduction, and unless one considers "closed" notions (such that one can absorb objects into the distinguisher without changing the performance), context insensitivity simply does not hold. The following lemma makes explicit in which way parallel resources affect the distinguishing performance. It is phrased with respect to a single resource T composed in parallel, however, the analogous statement holds with respect to tuples with arbitrarily many resources, and with respect to tuples with more general label sets.

Lemma 3.14. *Let R, S, and T be resources, and let $\boldsymbol{\pi}$ be a protocol, σ be a simulator and ε a performance measure such that*

$$R \quad \xRightarrow{\boldsymbol{\pi},\sigma,\varepsilon_{\boldsymbol{\pi}}} \quad S.$$

Then

$$(R,T) \quad \xRightarrow{\langle\boldsymbol{\pi}\rangle_1,\langle\sigma\rangle_1,\varepsilon} \quad (S,T)\,,$$

where $\varepsilon^1 = \varepsilon^1_{\boldsymbol{\pi}} \circ (\cdot, T_\perp)$ and $\varepsilon^2 = \varepsilon^2_{\boldsymbol{\pi}} \circ (\cdot, T)$.

Proof. By definition, $[\![(\boldsymbol{\pi}R_\perp \mid S_\perp)]\!] \leq \varepsilon^1_{\boldsymbol{\pi}}$ and $[\![(\boldsymbol{\pi}R \mid \sigma^E S)]\!] \leq \varepsilon^2_{\boldsymbol{\pi}}$. Consequently,

$$
\begin{aligned}
[\![(\langle\boldsymbol{\pi}\rangle_1 (R_\perp, T_\perp) \mid (S_\perp, T_\perp))]\!] &= [\![((\boldsymbol{\pi}R_\perp, T_\perp) \mid (S_\perp, T_\perp))]\!] \\
&= [\![((\cdot, T_\perp)\,\boldsymbol{\pi}R_\perp \mid (\cdot, T_\perp)\,S_\perp)]\!] \\
&= [\![(\boldsymbol{\pi}R_\perp \mid S_\perp)]\!] \circ (\cdot, T_\perp) \\
&= \varepsilon^1_{\boldsymbol{\pi}} \circ (\cdot, T_\perp)\,,
\end{aligned}
$$

using Lemma 2.6, and analogously

$$
\begin{aligned}
[\![(\langle\boldsymbol{\pi}\rangle_1 (R, T) \mid \langle\sigma\rangle_1 (S, T))]\!] &= [\![((\boldsymbol{\pi}R, T) \mid (\sigma S, T))]\!] \\
&= [\![((\cdot, T)\,\boldsymbol{\pi}R \mid (\cdot, T)\,\sigma S)]\!] \\
&= [\![(\boldsymbol{\pi}R \mid \sigma S)]\!] \circ (\cdot, T) \\
&= \varepsilon^2_{\boldsymbol{\pi}} \circ (\cdot, T)\,.
\end{aligned}
$$

This completes the proof. □

3.3.3 Parametrized Statements

Security statements about cryptographic protocols are often parametrized with values that can be parameters of the scheme, such as the key length of an encryption scheme, or parameters of the environment in which the protocol is used, such as the number of messages that are encrypted. The parameters determine the exact bound that can be proven for the cryptographic scheme. From a constructive perspective, this corresponds to the fact that the constructive statement is parametrized, that is, for a given parameter set $\mathcal{P}$, the construction holds for all parameters $p \in \mathcal{P}$, with a specific bound for each parameter.

In more detail, the components for a parametrized statement are families of (pairs of) resources,

$$\Omega_{\mathcal{P}} \quad := \quad \prod_{p \in \mathcal{P}} (\Phi \times \Phi),$$

and the constructors are families of triples

$$\Gamma_{\mathcal{P}} \quad := \quad \prod_{p \in \mathcal{P}} \left(\Pi \times \Sigma \times (\mathcal{D} \to [0,1])^2 \right),$$

where each triple consists of a protocol, a simulator, and a pair of performance measures. We refer to the objects as *parametrized* resources, protocols, simulators, and performance measures.

Definition 3.15 (Parametrized construction)**.** For a parameter set $\mathcal{P}$, let $R = \{R_p\}_{p \in \mathcal{P}}$ and $S = \{S_p\}_{p \in \mathcal{P}}$ be families of resources, $\boldsymbol{\pi} = \{\boldsymbol{\pi}_p\}_{p \in \mathcal{P}}$ a family of protocols, $\sigma = \{\sigma_p\}_{p \in \mathcal{P}}$ a family of simulators, and $\varepsilon = \{\varepsilon_p\}_{p \in \mathcal{P}}$ be a family of performance measures. Then $\boldsymbol{\pi}$ *constructs* S *from* R*, with respect to simulator* σ *and within* ε, if

$$R \quad \xRightarrow{\boldsymbol{\pi},\sigma,\varepsilon} \quad S \qquad :\Longleftrightarrow \qquad \forall p \in \mathcal{P} : R_p \quad \xRightarrow{\boldsymbol{\pi}_p,\sigma_p,\varepsilon_p} \quad S_p.$$

◇

We use the same "arrow symbol" as for the constructions from Definition 3.12 because those constructions can be seen as a special case where the families consist only of a single element. The composition theorem extends to this case, the statement follows since the condition is defined for each parameter individually, and Theorem 3.13 applies in each case.

Corollary 3.16. *For a parameter set* $\mathcal{P}$*, let* R*,* S *and* T *be parametrized resources as in Definition 3.15, and let* $\boldsymbol{\pi}$ *and* $\boldsymbol{\psi}$ *be parametrized protocols. Let*

$\sigma_{\boldsymbol{\pi}}$ and $\sigma_{\boldsymbol{\psi}}$ be parametrized simulators and $\varepsilon_{\boldsymbol{\pi}}$ and $\varepsilon_{\boldsymbol{\psi}}$ be suitable performance measures such that

$$R \xRightarrow{\boldsymbol{\pi},\sigma_{\boldsymbol{\pi}},\varepsilon_{\boldsymbol{\pi}}} S \quad \text{and} \quad S \xRightarrow{\boldsymbol{\psi},\sigma_{\boldsymbol{\psi}},\varepsilon_{\boldsymbol{\psi}}} T.$$

Then

$$R \xRightarrow{\boldsymbol{\psi}\circ\boldsymbol{\pi},\sigma_{\boldsymbol{\pi}}\circ\sigma_{\boldsymbol{\psi}},\varepsilon} T,$$

where $\varepsilon^1_p = \varepsilon^1_{\boldsymbol{\pi},p} \circ (\cdot\boldsymbol{\psi}_p) + \varepsilon^1_{\boldsymbol{\psi},p}$ and $\varepsilon^2_p = \varepsilon^2_{\boldsymbol{\pi},p} \circ (\cdot\boldsymbol{\psi}_p) + \varepsilon^2_{\boldsymbol{\psi},p} \circ (\cdot{\sigma_{\boldsymbol{\pi},p}}^E)$ for each $p \in \mathcal{P}$.

(Static) corruption of parties. In settings with multiple parties, one is often interested in statements about scenarios in which one or more of the parties are *corrupted* by an attacker. Here, corruption refers to a setting in which the attacker has full control over the (computer of the) party, e.g. by infecting the computer with malware, and a security statement that takes such corruptions into account formalizes the guarantees that can still be given to the remaining uncorrupted parties. Security statements with respect to *static* corruptions, that is, the corrupted parties are fixed in advance and cannot be chosen during the protocol execution, can be formalized as parametrized statements, where the parameter is the set of corrupted parties.

Constructions with respect to static corruptions are formalized by considering the parameter set $\mathcal{P} := 2^{(\mathcal{I}\setminus\{E\})}$, where the parameter $\mathcal{I}_{\mathsf{C}} \in \mathcal{P}$ describe the setting in which the parties at interface $I \in \mathcal{I}_{\mathsf{C}}$ are considered corrupted. A resource R in this setting is (formally) a family of resources $\{R_{\mathcal{I}_{\mathsf{C}}}\}_{\mathcal{I}_{\mathsf{C}}\in\mathcal{P}}$, where the resource $R_{\mathcal{I}_{\mathsf{C}}}$ will (usually) be described as providing the capabilities corresponding to the interfaces $I \in \mathcal{I}_{\mathsf{C}}$ at the E-interface.

Following Definition 3.15, constructing a resource S from a resource R with respect to static corruption then means that the condition must hold with respect to each set $\mathcal{I}_{\mathsf{C}} \subseteq \mathcal{I}$. More formally, the construction statement is described with respect to a family $\sigma = \{\sigma_{\mathcal{I}_{\mathsf{C}}}\}_{\mathcal{I}_{\mathsf{C}}\in\mathcal{P}}$ and pairs of performance measures $\varepsilon = \{\varepsilon_{\mathcal{I}_{\mathsf{C}}}\}_{\mathcal{I}_{\mathsf{C}}\in\mathcal{P}}$ such that

$$\forall \mathcal{I}_{\mathsf{C}} \subseteq \mathcal{I}\setminus\{E\}: \quad R_{\mathcal{I}_{\mathsf{C}}} \xRightarrow{\boldsymbol{\pi}_{\mathcal{I}_{\mathsf{C}}},\sigma_{\mathcal{I}_{\mathsf{C}}},\varepsilon_{\mathcal{I}_{\mathsf{C}}}} S_{\mathcal{I}_{\mathsf{C}}},$$

where $\boldsymbol{\pi}_{\mathcal{I}_{\mathsf{C}}}$ consists of the converters for all $I \in \mathcal{I}\setminus(\{E\}\cup\mathcal{I}_{\mathsf{C}})$. The protocol $\boldsymbol{\pi}$ is then said to *construct S from R with respect to static corruptions* if the above condition is satisfied.

3.4 Discrete Systems

Most cryptographic protocols consist of *discrete* systems in the sense that the behavior of each protocol engine is described as a step-wise process where

in each step the protocol engine takes as inputs values in some specified set, performs some computation, and provides as outputs also values in some set. A protocol execution is then formalized as an interaction of several such discrete systems.

Random systems as defined by Maurer [Mau02] and also described in Section 2.3 formalize discrete behavior as a sequence of conditional probability distributions of outputs $Y_1, Y_2, \ldots$ given inputs $X_1, X_2, \ldots$, where in each step the system takes an input X_i and provides an output Y_i. This formalization, however, is too restrictive to capture the behavior of arbitrary discrete systems because random systems are only defined if the inputs are provided in the predetermined order $X_1, X_2, \ldots$, whereas for an arbitrary discrete system, the order in which the inputs are given is not fixed in advance.

In Section 3.4.1, we specify a model of discrete systems which captures this more general type of behavior. The model can be seen as an extension of random systems in the sense that the order of inputs is not fixed in advance, it can also be seen as a modification of Kahn networks [Kah74] to support probabilistic behavior. We prove that this type of discrete system fulfills the axioms of the abstract system algebra described in Section 3.2, and show how the distinguishing advantage appears in this context. In Section 3.4.2, we show how statements about discrete systems can be parametrized such that they capture parameters such as the number of transmitted messages without having the protocol depend on this parameter; this uniformity is necessary if no upper bound is known in advance and hence the protocol must work for an *a priori* unbounded number of messages. We then describe in Section 3.4.3 how reduction proofs following the ideas described in Section 2.2 will be performed in the subsequent chapters. Finally, in Section 3.4.4, we introduce a specification language that we use for describing discrete systems in Chapters 4 and 5.

3.4.1 The Algebra of Monotone Discrete Systems

Generally, a discrete system has a fixed set of input interfaces at which it accepts input values, and a set of output interfaces at which it provides output values. Each input or output interface corresponds to a unique (potential) input or output value in some value space $\mathcal{X}$, and is labeled by an element of some label set $\mathcal{J}$. The behavior of the system is described as a sequence of activations, where in each activation the system receives one or more inputs and potentially provides one or more outputs. The order in which the inputs are given at the input interfaces is not fixed in advance; the system must handle arbitrary orders.

If the system behavior is not restricted and may in particular depend on

the order in which the inputs are given—in contrast to only depending on the values of the inputs—then the composition of multiple systems with several connected interfaces introduces non-determinism. The reason is that if multiple inputs without an explicit order are provided to a system, and the behavior of the system depends on the order in which the inputs are given, then the output values are not well-defined. From a different perspective, this can be viewed as an impossibility of achieving a modular description of a composite system and is usually referred to as the Brock-Ackermann anomaly [BA83]. A detailed exposition of these effects has been given by Brock [Bro83].

For a specific sub-type of systems, however, these problems do not arise; these are the systems for which the behavior does not depend on the order in which the inputs are given; as in the model of Kahn [Kah74]. In the deterministic case, this type of system can be described by a function which maps tuples of input values to tuples of output values. As a tuple does not encode the order, such a description of behavior is necessarily independent of the order.

The formal type of discrete systems. If a system is described by a mapping from tuples of inputs to tuples of outputs, and each activation of the system corresponds to an evaluation of the function on the partial tuple that describes all inputs provided so far, then the function must be monotone in the sense that "more defined" inputs will lead to "more defined" outputs. This monotonicity assures that the behavior of the system remains consistent over time; when further inputs are given, the output of the function is consistent with the values of the outputs provided in previous steps. The monotonicity is formalized with respect to the following order relation on (partial) tuples.

Definition 3.17. Let $\mathcal{X}$ be some set, let $\mathcal{J}$ be a discrete set, and let $x, \bar{x} \in \mathcal{X}^{(\mathcal{J})}$. Then x *is more defined than* $\bar{x}$, or $x \geq \bar{x}$, if for all $j \in \operatorname{supp} \bar{x}$ we have that $\bar{x}_j = x_j$. Analogously, x *is strictly more defined than* $\bar{x}$, or $x > \bar{x}$, if $x \geq \bar{x}$ and $x \neq \bar{x}$. ◇

Following the above description, the behavior of a *deterministic* discrete system whose output does not depend on the order of inputs is formally specified by a function

$$f: \quad \mathcal{X}^{(\mathcal{J})} \rightarrow \mathcal{X}^{(\mathcal{J})}$$

which is *monotone* with respect to the order relation from Definition 3.17. For each such function f, we define the sets of (potential) input and output interfaces, denoted $\mathrm{In}(f)$ and $\mathrm{Out}(f)$, as the smallest subsets of $\mathcal{J}$ such that f can be described as a function $\mathcal{X}^{(\mathrm{In}(f))} \rightarrow \mathcal{X}^{(\mathrm{Out}(f))}$. In other words, $\mathrm{In}(f)$ contains the labels in $\mathcal{J}$ for which an input at the corresponding position of

the tuple potentially affects the function value, and $\mathrm{Out}(f)$ contains the labels in $\mathcal{J}$ for which there exists an input to f such that the output tuple contains a value at the respective position. A function is *finitary* if both sets $\mathrm{In}(f)$ and $\mathrm{Out}(f)$ are finite.

Probabilistic behavior can be described by considering a random function, i.e., a random variable F over functions $f : \mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$. Such a random function F then defines an $\big(\mathcal{X}^{(\mathcal{J})}, \mathcal{X}^{(\mathcal{J})}\big)$-random system $\mathbf{F}$ as follows:

$$\mathsf{p}^{\mathbf{F}}_{Y_k|X^kY^{k-1}}(y_k, x^k, y^{k-1}) \coloneqq \mathsf{P}(F(x_k) = y_k \mid \forall l < k : F(x_l) = y_l). \quad (3.14)$$

Note that several random functions which correspond to different random variables may still result in the same random system. One example is the system that takes as input a single bit, and provides as output a uniformly random bit. This system can be decomposed either into "output 0" and "output 1" with probability $1/2$ each, or into "keep" and "flip," again with probability $1/2$ each.

We define monotone discrete systems as random systems that are induced by monotone random functions as described in equation (3.14); equivalently, they can be seen as equivalence classes if such random functions. The definitions of $\mathrm{In}(\cdot)$ and $\mathrm{Out}(\cdot)$ extend to this case.

Definition 3.18. Let $\mathcal{X}$ be some set and $\mathcal{J}$ be a label set. A *monotone discrete system* is a random system $\mathbf{F}$ that is induced by a monotone random function, that is, a random variable over functions $\mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$ according to equation (3.14). $\diamond$

We define operations on the systems on their descriptions as monotone functions, the operations extend to probabilistic systems by defining them on the realizations of the random variable.

Parallel composition. The parallel composition of monotone functions is defined as evaluating each function independently of the other functions. For functions f and f' with $\mathrm{In}(f) \cap \mathrm{In}(f') = \mathrm{Out}(f) \cap \mathrm{Out}(f') = \emptyset$, the *parallel composition of f and f'* is the function

$$\begin{aligned} f \parallel f' : \quad \mathcal{X}^{(\mathcal{J})} &\to \mathcal{X}^{(\mathcal{J})} \\ x \cup x' &\mapsto f(x) \cup f(x'), \end{aligned}$$

where $\operatorname{supp} x \subseteq \mathrm{In}(f)$ and $\operatorname{supp} x' \subseteq \mathrm{In}(f')$. It is easy to see that $\mathrm{In}(f \parallel f') = \mathrm{In}(f) \cup \mathrm{In}(f')$, and $\mathrm{Out}(f \parallel f') = \mathrm{Out}(f) \cup \mathrm{Out}(f')$. The operation extends to monotone discrete systems, as shown in the following lemma.

Lemma 3.19. *For two monotone discrete systems $\mathbf{F}, \mathbf{F}'$, the composition $\mathbf{F} \parallel \mathbf{F}'$ is well-defined. Concretely, let F, F' and G, G' be random variables over functions $\mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$ such that F, G are both contained in the equivalence class $\mathbf{F}$ and F', G' are both in the equivalence class $\mathbf{F}'$. Then $F \parallel F'$ and $G \parallel G'$ are equivalent.*

Proof. The statement can be shown by simply computing the probabilities. For all sequences $x_1 \leq x_2 \leq \cdots \leq x_k, x'_1 \leq x'_2 \leq \cdots \leq x'_k \in \mathcal{X}^{(\mathcal{J})}$ with $\operatorname{supp} x_k \subseteq \mathrm{In}(F)$ and $\operatorname{supp} x'_k \subseteq \mathrm{In}(F')$, and $y_1 \leq y_2 \leq \cdots \leq y_k, y'_1 \leq y'_2 \leq \cdots \leq y'_k \in \mathcal{X}^{(\mathcal{J})}$ with $\operatorname{supp} y_k \subseteq \mathrm{Out}(F)$ and $\operatorname{supp} y'_k \subseteq \mathrm{Out}(F')$,

$$\begin{aligned}
&\mathsf{P}\left(\forall l : (F \parallel F')(x_l \cup x'_l) = y_l \cup y'_l\right) \\
&\qquad = \mathsf{P}\left(\forall l : F(x_l) = y_l\right) \cdot \mathsf{P}\left(\forall l : F'(x'_l) = y'_l\right) \\
&\qquad = \mathsf{P}\left(\forall l : G(x_l) = y_l\right) \cdot \mathsf{P}\left(\forall l : G'(x'_l) = y'_l\right) \\
&\qquad\qquad = \mathsf{P}\left(\forall l : (G \parallel G')(x_l \cup x'_l) = y_l \cup y'_l\right),
\end{aligned}$$

where the first and last step follow since the random variables are independent and the intermediate step uses the equivalence in the individual cases. □

Connecting interfaces. The interface-connecting operation is described by a pair $(i, j) \in \mathcal{J} \times \mathcal{J}$ of labels with the interpretation that the output y_j is identified with the input x_i. Intuitively, applying the operation described by (i, j) to a system results in a "reduced" system where the variables y_j and x_i are no longer accessible from the outside.[7]

To formally define this operation, we introduce some new notation. First, for a tuple $x \in \mathcal{X}^{(\mathcal{J})}$ and $j \in \mathcal{J}$, we denote by $x \setminus j$ the tuple that coincides with x at all labels except for j and is undefined at label j. (This notation is motivated by the fact that if one views the tuple as a set of pairs, then the operation corresponds to removing the pair with the first component equal to j from the set.) Second, for a tuple $x \in \mathcal{X}^{(\mathcal{J})}$, $j \in \mathcal{J}$, and $y \in \mathcal{X}$, we denote by $x \mid j \mapsto y$ the tuple that coincides with x at all labels except for j and has the value y at position j. (This notation is motivated by the fact that the tuple is the same as x except for label j, which "maps" to y.) Third, for an label j, we denote by $f_j(x)$ the value of the j-component of the tuple $f(x)$.

Then the new system is described by a function f' with $\mathrm{In}(f') = \mathrm{In}(f) \setminus \{i\}$ and $\mathrm{Out}(f') = \mathrm{Out}(f) \setminus \{j\}$. More formally, the system is described by the function

$$\gamma_{i:j}(f)(x) \quad := \quad f(x \mid i \mapsto f_j(x)) \setminus j.$$

[7] If an output (or event) shall be visible to more than one other system, the "sending" system will simply define several outputs that always have a consistent value.

It is easy to see that $\mathrm{In}(\gamma_{i:j}(f)) = \mathrm{In}(f) \setminus \{i\}$ and $\mathrm{Out}(\gamma_{i:j}(f)) = \mathrm{Out}(f) \setminus \{j\}$. This operation is also compatible with the equivalence classes, as shown in the following lemma..

Lemma 3.20. *For a monotone discrete system* $\mathbf{F}$*, the system* $\gamma_{i:j}(\mathbf{F})$ *is well-defined. Concretely, let* F *and* G *be random variables over functions* $\mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$ *such that* F *and* G *are both contained in the equivalence class described by* $\mathbf{F}$*. Then* $\gamma_{i:j}(F)$ *and* $\gamma_{i:j}(G)$ *are equivalent.*

Proof. The statement can again be shown by computing the probabilities. Let $x_1, x_2, \ldots, x_k \in \mathcal{X}^{(\mathrm{In}(\mathbf{F}))}$ and $y_1, y_2, \ldots, y_k \in \mathcal{X}^{(\mathrm{Out}(\mathbf{F}))}$. Recall that

$$\gamma_{i:j}(f)(x) = f(x \mid i \mapsto f_j(x)) \setminus j,$$

with which we obtain:

$$\begin{aligned}
&\mathsf{P}(\forall l : \gamma_{i:j}(F)(x_l) = y_l) \\
&\quad = \mathsf{P}(\forall l : F(x_l \mid i \mapsto F_j(x_l)) \setminus j = y_l) \\
&\quad = \sum_{\substack{t \in \mathcal{X}, \\ l' \leq n}} \mathsf{P}\big((\forall l < l' : F(x_l) = y_l) \wedge (F_j(x_l) = t) \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad (\forall l \geq l' : F(x_l \mid i \mapsto t) \setminus j = y_i)\big) \\
&\qquad + \mathsf{P}(\forall l : F(x_l) = y_l) \\
&\quad = \sum_{\substack{t \in \mathcal{X}, \\ l' \leq n}} \mathsf{P}\big((\forall l < l' : G(x_l) = y_l) \wedge (G_j(x_l) = t) \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad (\forall l \geq l' : G(x_l \mid i \mapsto t) \setminus j = y_l)\big) \\
&\qquad + \mathsf{P}(\forall l : G(x_l) = y_l) \\
&\quad = \mathsf{P}(\forall l : G(x_l \mid i \mapsto G_j(x_l)) \setminus j = y_l) \\
&\quad = \mathsf{P}(\forall l : \gamma_{i:j}(G)(x_l) = y_l),
\end{aligned}$$

where the step in the middle follows from the equivalence. □

Relabeling interfaces. For a full instantiation of the system algebra in Definitions 3.4 and 3.5, we also need to describe how interfaces are relabeled for discrete monotone systems. For a function $f : \mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$ and a pair of injective mappings $\rho_{\mathrm{in}}, \rho_{\mathrm{out}} : \mathcal{J} \to \mathcal{J}$, the function $(\rho_{\mathrm{in}}, \rho_{\mathrm{out}})(f) : \mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$ is defined as follows. For an input tuple $x \in \mathcal{X}^{(\rho_{\mathrm{in}}(\mathrm{In}(f)))}$, first the labels of x are modified via ρ_{in}^{-1}, then the function f is applied, and finally the labels of the obtained tuple are modified via ρ_{out} to obtain a tuple in $\mathcal{X}^{(\rho_{\mathrm{out}}(\mathrm{Out}(f)))}$.

More formally, one can understand a function $\rho : \mathcal{J} \to \mathcal{J}$ as a mapping $\mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$ via

$$\rho(x) \quad \coloneqq \quad \{(\rho(j), t) : (j, t) \in x\},$$

where x is understood as a set of pairs, i.e., a subset of $\mathcal{J} \times \mathcal{X}$. With this operation, relabeling the function f can be formally defined as

$$(\rho_{\text{in}}, \rho_{\text{out}})(f) \quad := \quad \rho_{\text{out}} \circ f \circ \rho_{\text{in}}^{-1}.$$

The definition extends to random variables over $\mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$ and equivalence classes in the straightforward way. Of course, $\text{In}((\rho_{\text{in}}, \rho_{\text{out}})(f)) = \rho_{\text{in}}(\text{In}(f))$ and $\text{Out}((\rho_{\text{in}}, \rho_{\text{out}})(f)) = \rho_{\text{out}}(\text{Out}(f))$.

Defining the algebra of monotone discrete systems. We prove that the algebra of monotone discrete systems is indeed a connection-order invariant system algebra. As a result, we can use it as an underlying formalism for constructive security statements. To describe monotone discrete systems as a system algebra, we denote the input interfaces by elements of $\mathcal{J} \times \{?\}$ and the output interfaces by elements of $\mathcal{J} \times \{!\}$. In more detail, the input interface i of a monotone discrete system is denoted as $i?$ in the algebra, and the output interface j of a monotone discrete system is denoted as $j!$ in the algebra. Thus, the system algebra has interface labels in the set $\mathcal{J} \times \{?, !\}$. The interface-connection operation is only partially defined, namely for cases where one input and one output interface are connected.

Theorem 3.21. *Let $\mathcal{X}$ and $\mathcal{J}$ be as above. Then the monotone discrete systems with the above described operations define a connection-order independent system algebra with interfaces labels in the set $\mathcal{J} \times \{?, !\}$.*

Proof. It is obvious that the operations, when applied to finitary functions, result in finitary functions. We show that the operations also preserve monotonicity. Indeed, for the parallel composition, for monotone functions $f_1, f_2 : \mathcal{X}^{(\mathcal{J})} \to \mathcal{X}^{(\mathcal{J})}$, and $x_1, x_2, x_2', x_2' \in \mathcal{X}^{(\mathcal{J})}$ with $\operatorname{supp} x_1 \subseteq \operatorname{supp} x_1' \subseteq \text{In}(f_1)$, $\operatorname{supp} x_2 \subseteq \operatorname{supp} x_2' \subseteq \text{In}(f_2)$, $x_1 \leq x_1'$, and $x_2 \leq x_2'$,

$$(f_1 \parallel f_2)(x_1 \cup x_2) \;=\; f_1(x_1) \cup f_2(x_2) \;\leq\; f_1(x_1') \cup f_2(x_2') \;=\; (f_1 \parallel f_2)(x_1' \cup x_2')$$

and hence $f_1 \parallel f_2$ is also monotone.

The function $\gamma_{i:j}(f)$ is also monotone: for two inputs $x, x' \in \mathcal{X}^{(\mathcal{J})}$ with $x \leq x'$ and $\operatorname{supp} x' \subseteq \text{In}(f)$, the monotonicity of f implies that $f_j(x) \leq f_j(x')$, consequently $x \mid i \mapsto f_j(x) \quad \leq \quad x' \mid i \mapsto f_j(x')$ and again by the monotonicity of f we conclude $\gamma_{i:j}(f)(x) \leq \gamma_{i:j}(f)(x')$.

To show that the algebra is indeed connection-order invariant, we have to show that the order in which two feedbacks are applied does not matter. More concretely, the condition means that applying (i, j) and (i', j')—with $i \neq i'$ and $j \neq j'$—in any order has the same effect:

$$f'(x) = f(x \mid i \mapsto f_j(x)) \setminus j,$$

so for any $x \in \mathcal{X}^{(\mathrm{In}(f)\setminus\{i,i'\})}$,

$$\begin{aligned} f''(x) &= f'\left(x \mid i' \mapsto f'_{j'}(x)\right) \setminus j' \\ &= f\left(x \mid i' \mapsto f'_{j'}(x), i \mapsto f_j(x \mid i' \mapsto f'_{j'}(x))\right) \setminus j, j' \\ &= f\left(x \;\middle|\; \begin{array}{l} i' \mapsto f_{j'}\left(x \mid i \mapsto f_j(x)\right), \\ i \mapsto f_j\left(x \mid i' \mapsto f_{j'}\left(x \mid i \mapsto f_j(x)\right)\right) \end{array}\right) \setminus j, j' \end{aligned}$$

and analogously

$$\bar{f}'(x) = f\left(x \mid i' \mapsto f_{j'}(x)\right) \setminus j',$$

which results in

$$\begin{aligned} \bar{f}'' &= \bar{f}'(x \mid i \mapsto \bar{f}'_j(x)) \setminus j \\ &= f\left(x_| i \mapsto \bar{f}'_j(x), i' \mapsto f_{j'}(x \mid i \mapsto \bar{f}'_j(x))\right) \setminus j, j' \\ &= f\left(x \;\middle|\; \begin{array}{l} i \mapsto f_j\left(x \mid i' \mapsto f_{j'}(x)\right), \\ i' \mapsto f_{j'}(x \mid i \mapsto f_j\left(x \mid i' \mapsto f_{j'}(x)\right)) \end{array}\right) \setminus j, j'. \end{aligned}$$

Hence, we have to show that $f'' = \bar{f}''$ (which is trivial for the labels j and j'). The general case is as follows. Consider how i' is set for the outermost evaluation of f: if $f_{j'}(x) = \square$, then the innermost $f_j(\dots)$ is evaluated on input x both in f'' and in $\bar{f}''$, hence it returns the same result and i' is set consistently in both cases. If $f_{j'}(x) \neq \square$, then i' is set consistently by monotonicity. The analogous argument holds for i.

The consistency with the relabeling mappings is easy to verify. $\square$

Distinguishers for discrete monotone systems and the distinguishing advantage. As the discrete monotone systems are shown to fulfill the axioms of the abstract system algebra from Definition 3.4, the results of Sections 3.2.3 and 3.3 lead to a construction notion between resources which are described by (pairs of) discrete monotone systems and which is based on the distinction problem between the systems that describe the resources. To apply these definitions, we have to describe how the corresponding distinguishing advantage is defined.

Following the construction of a cryptographic algebra from any system algebra in Section 3.2.3, a resource in a cryptographic algebra with interface labels in $\mathcal{I}$ is a system with interfaces labeled by elements in the set $\mathcal{J} = \mathcal{I} \times \mathcal{L}$ for some set $\mathcal{L}$. As described in Section 3.4.1, such a discrete monotone system can be viewed as an $\left(\mathcal{X}^{(\mathcal{J})}, \mathcal{X}^{(\mathcal{J})}\right)$-random system. The distinction problem and distinguishing advantage for discrete monotone systems is then defined by the notions for these random systems as described in Definition 2.10.

More formally, the distinction advantage monotone discrete resources is defined based as the advantage in terms of random systems, and the set $\mathcal{D}$ of distinguishers for discrete monotone systems is the set of $\left(\mathcal{X}^{(\mathcal{J})}, \mathcal{X}^{(\mathcal{J})}\right)$-distinguishers according to Definition 2.10.

Definition 3.22. Let $\mathbf{R}$ and $\mathbf{S}$ be discrete monotone resources with interface labels in $\mathcal{J} = \mathcal{I} \times \mathcal{L}$ with an MBO that determines when the resource "stops." Let $\mathbf{D}$ be an $\left(\mathcal{X}^{(\mathcal{J})}, \mathcal{X}^{(\mathcal{J})}\right)$-distinguisher as in Definition 2.10. Then the *distinguishing advantage of* $\mathbf{D}$ *in distinguishing* $\mathbf{R}$ *and* $\mathbf{S}$ is defined as

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \quad := \quad \left|\mathsf{P}^{\mathbf{DR}}(W = 1) - \mathsf{P}^{\mathbf{DS}}(W = 1)\right|,$$

that is, as the distinguishing advantage as random systems. ◇

According to Definition 2.10, the random variable W is defined once a special MBO of the connected system ($\mathbf{R}$ or $\mathbf{S}$) becomes 1. We describe in Section 3.4.2 how this definition is used to formalize parametrized statements.

For a performance function $\varepsilon = [\![(\mathbf{R} \mid \mathbf{S})]\!]$, terms of the form $\varepsilon \circ (\cdot\alpha^I)$ appear in the composition theorem. This is, by definition,

$$\left(\varepsilon \circ (\cdot\alpha^I)\right)(\mathbf{D}) \quad = \quad [\![\left(\alpha^I\mathbf{R} \mid \alpha^I\mathbf{S}\right)]\!](\mathbf{D}) \quad = \quad \Delta^{\mathbf{D}}\left(\alpha^I\mathbf{R}, \alpha^I\mathbf{S}\right),$$

for all distinguishers $\mathbf{D}$. Analogously, one can view this of the advantage of a distinguisher $\mathbf{D}\alpha^I$ in distinguishing $\mathbf{R}$ from $\mathbf{S}$, where the advantage is defined as first composing α at interface I to $\mathbf{R}$ and $\mathbf{S}$, and then computing the advantage of $\mathbf{D}$ on the obtained resources. With this more general interpretation of distinguishers, composition-order invariance holds by definition of the composition.

3.4.2 Parametrized Statements and Uniformity

For several types of cryptographic schemes, one is interested in providing security bounds that depend on how often the scheme is used, or on the amount of resources a potential attacker may have. For instance, in a setting where messages are authenticated by "authentication tags" generated by a uniform random function, the probability of the attacker to simply guess a valid tag, and hence forge a message, clearly depends on the number of attempts the attacker can make. The protocol is required to not depend on the parameters, because the restriction on the attacker (or often even the number of messages that shall be transmitted during the execution of the protocol) is *a priori* not bounded. The purpose of the parameters is to make the *analysis* of the scheme dependent on the use of the scheme. In this sense, we are interested in a "uniform" parametrization in which each object (like resources and protocols) is described as a parametrization of the same discrete monotone system.

To faithfully capture such a parametrization in terms of such "uniform" parameters, we specialize the parametrized constructions described in Section 3.3.3. The families of protocols and resources are each described by a single monotone discrete system, and for each parameter $p \in \mathcal{P}$, the element of the family is obtained by extending the discrete system by a special monotone binary output (MBO). The distinction advantage is then defined by using the disjunction of all these MBOs (the resources and the protocol resp. simulator may all have such an MBO) to determine when the random experiment stops, i.e., which output of the distinguisher is used as the random variable W according to Definition 2.5. This formalization assures that the protocol and resources are *uniform* in the sense that their behavior is independent of the parameter, the MBOs are only introduced to make the security statement and determine the end of the random experiment.

More formally, this means that a (uniform) parametrized construction with respect to discrete systems is defined on a parametrized component set $\Omega_{\mathcal{P}}$ and a parametrized constructor set $\Gamma_{\mathcal{P}}$ as in Section 3.3.3, where each family in these parametrized sets is obtained by parametrizing a monotone discrete system with a family of MBOs. Combining Definitions 2.10 and 3.15, a parametrized statement is then defined as follows.

Definition 3.23. Let $\mathcal{P}$ be some set of parameters. Let $(\mathbf{R}, \mathbf{R}_\perp)$ and $(\mathbf{S}, \mathbf{S}_\perp)$ be resources, $\boldsymbol{\pi}$ be a protocol, and σ be a simulator, which are all monotone discrete systems. For each $p \in \mathcal{P}$, consider an MBO on each of the systems (resulting in systems $\mathbf{R}^p$, $\mathbf{R}^p_\perp$, $\mathbf{S}^p$, $\mathbf{S}^p_\perp$, σ^p and a tuple of systems $\boldsymbol{\pi}^p$ which all have MBOs).

For a family of (pairs of) performance measures $\varepsilon^1_{\mathcal{P}} = \{\varepsilon^1_p\}_{p\in\mathcal{P}}$ and $\varepsilon^2_{\mathcal{P}} = \{\varepsilon^2_p\}_{p\in\mathcal{P}}$, the *protocol* $\boldsymbol{\pi}$ *constructs* $(\mathbf{S}, \mathbf{S}_\perp)$ *from* $(\mathbf{R}, \mathbf{R}_\perp)$ *with respect to* $\sigma_{\mathcal{P}}$ *and within* $\varepsilon_{\mathcal{P}} = \left(\varepsilon^1_{\mathcal{P}}, \varepsilon^2_{\mathcal{P}}\right)$, *with parameter set* $\mathcal{P}$, if, for all $p \in \mathcal{P}$:

$$[\![(\boldsymbol{\pi}^p\mathbf{R}^p_\perp \mid \mathbf{S}^p_\perp)]\!] \quad \leq \quad \varepsilon^1_p$$

and

$$\left[\!\!\left[\left(\boldsymbol{\pi}^p\mathbf{R}^p \mid (\sigma^p)^E\mathbf{S}^p\right)\right]\!\!\right] \quad \leq \quad \varepsilon^2_p.$$

◇

Here, the distinction advantage is defined as described in Definition 2.10, i.e., the output of the distinguisher is determined once the MBO becomes 1.

As a specific notation, if for two discrete systems $\mathbf{R}$ and $\mathbf{S}$ with MBOs that determines when the distinguishing experiment stops, we write $\mathbf{R} \equiv \mathbf{S}$, then this is to be understood as $\mathbf{R} \overset{\mathrm{g}}{\equiv} \mathbf{S}$ with respect to those MBOs. The reason for this notation is that we will make statements in settings where MBOs appear

both in the sense of "winning a game" and in the sense of "restricting the use," and the specific notation shall clarify to which of the MBOs we refer.

3.4.3 Discrete Games and Reductions

Security statements about protocols which are based on computational assumptions are formalized by providing an explicit reduction from the computational problem formalizing the assumption to breaking the security of the scheme. The computational problems are specified in terms of (discrete) games as described in Section 2.4.1, and a reduction maps a distinguisher for the security statement to a solver for such a game. In this section, we describe how the discrete games can be embedded into the system algebra developed in this chapter, and how reduction proofs are formalized.

Games such as those described in Section 2.4.1 either correspond to *abstract games* as in Definition 2.3 (if they formalize that a certain condition is hard to provoke, such as the unforgeability of a MAC or a signature scheme) or to *distinction problems* as in Definition 2.5 (if they formalize that two situations are difficult to tell apart, such as for the confidentiality of an encryption scheme).

The interfaces of a game according to Section 2.4.1 that is used by the game winner or distinguisher is described by certain procedures that can be called. For simplifying the formalization of the reductions, we slightly generalize the interpretation of the games given in Section 2.4.1 to allow, in each step, to provide a variable-length sequence of procedure calls. Consequently, as a random system (i.e., a family $\mathsf{p}_{Y_i|X^iY^{i-1}}$ of conditional probability distributions) the system takes as each input X_i zero or more calls to the procedures, and replies in Y_i by the corresponding results. Consistently with Section 2.4.1, we assume that some specific encoding of the sequences and procedure calls exists, each X_i contains a (variable length) list of elements which contain the identifier of the procedure. The result Y_i is a (variable length) list of elements which contain the return values of the corresponding procedures.

In a reduction from game-winning for a certain game to a distinction problem that arises in our security definitions, we have to describe a function that maps each distinguisher to a game winner and analyze how the performance is translated. Recall that a distinguisher is an environment for a discrete resource as described in Section 3.4.1; in particular, it can be viewed as an $\left(\mathcal{X}^{(\mathcal{J})}, \mathcal{X}^{(\mathcal{J})}\right)$-random system that in each step provides a tuple of messages to be given to the resource it is connected to, and as a result expects a tuple of outputs.

A reduction in this setting is also a sequence of conditional probability

distributions. In each step, it first obtains an input in the set $\mathcal{X}^{(\mathcal{J})}$ from the distinguisher and outputs a (variable length) list of procedure queries to the connected game. Second, it obtains a (variable length) list of return values from the game and outputs an element from the set $\mathcal{X}^{(\mathcal{J})}$ to the distinguisher. Abstractly, such a reduction can be seen as a converter which itself is a discrete system and connects with its inside interface to the system formalizing the computational problem, and with its outside interface to the distinguisher.

Definition 3.24. A (discrete) reduction system $\mathbf{C}$ is a family of conditional probability distributions

$$\left\{ \mathsf{p}^{\mathbf{C}}_{\bar{X}_i|X^i\bar{X}^{i-1}Y^{i-1}\bar{Y}^{i-1}}, \mathsf{p}^{\mathbf{C}}_{Y_i|X^i\bar{X}^iY^{i-1}\bar{Y}^i} \right\}_{i\geq 1},$$

where the random variables $X_1, X_2, \ldots$ and $Y_1, Y_2, \ldots$ take on values in the set $\mathcal{X}^{(\mathcal{J})}$, the random variables $\bar{Y}_1, \bar{Y}_2, \ldots$ describe sequences of procedure calls, and the random variables $\bar{X}_1, \bar{X}_2, \ldots$ describe sequences of results. $\diamond$

A random experiment with a distinguisher $\mathbf{D}$ (which is an environment in the sense of Definition 2.9 with inputs and outputs in $\mathcal{X}^{(\mathcal{J})}$), a reduction system $\mathbf{C}$, and a game $\mathbf{G}$ (which is a discrete game that takes as input sequences of procedure calls and provides as output sequences of results) is then described by the random experiment $\mathbf{DCG}$ obtained as

$$\mathsf{P}^{\mathbf{DCG}}_{X^iY^i\bar{X}^i\bar{Y}^i} \quad = \quad \mathsf{p}^{\mathbf{D}}_{X^i|Y^{i-1}} \cdot \mathsf{p}^{\mathbf{C}}_{\bar{X}^i|X^iY^{i-1}\bar{Y}^{i-1}} \cdot \mathsf{p}^{\mathbf{C}}_{Y^i|X^i\bar{X}^i\bar{Y}^i} \cdot \mathsf{p}^{\mathbf{G}}_{\bar{Y}^i|\bar{X}^i}.$$

The structure of the random experiment is indicated in Figure 3.5: Iteratively, $\mathbf{D}$ generates a value X_i, then $\mathbf{C}$, given this value, issues a query $\bar{X}_i$ to $\mathbf{G}$. After $\mathbf{G}$ responds with a value $\bar{Y}_i$, $\mathbf{C}$ provides the value Y_i to $\mathbf{D}$.

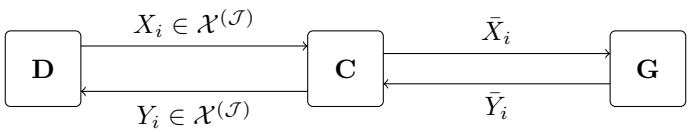


Figure 3.5: The setting with $\mathbf{D}$, $\mathbf{C}$, and $\mathbf{G}$.

3.4.4 Specification of Discrete Systems

The systems considered in the remainder of this thesis are monotone discrete systems as described in Section 3.4.1. Specifying the systems in terms of their mathematical type (i.e., the conditional probability distributions) is, however,

cumbersome and makes the behavior of the systems difficult to understand. In principle, any specification language that *uniquely* determines a system can be used to describe it. In this section, we describe a pseudo-code based description language that we will use in the remainder of the thesis, which allows for both a precise and comprehensible description, and for verifying that the described system is indeed monotone.

Recall that each system has a specific set of input and output interfaces which are specified by labels, and for resources with interfaces in $\mathcal{I}$. These inputs and outputs are identified by terms of pairs $(I, \boldsymbol{\lambda}) \in \mathcal{I} \times \boldsymbol{\Lambda}$ (where again $\boldsymbol{\Lambda} = \Lambda^*$ and the last character of the string $\boldsymbol{\lambda}$ is supposed to contain the information whether the interface is an input or output interface, hence it is in the set $\{?, !\}$).

We have defined monotone discrete systems with respect to a single value space $\mathcal{X}$ for all inputs and outputs of the systems. In the descriptions of specific systems, we will describe the variables with concrete value spaces; this can be seen as taking the set $\mathcal{X}$ to be the union of all value spaces used in the descriptions of the systems, and having the systems ignore values which are not in the set for which they are specified. Consequently, input values that are outside of the specified sets do not influence the behavior of the system; the system treats that as if the input was undefined.

Example: Insecure channel. The single-use insecure channel $- \rightarrow$ can be described as a system that only has a single input (called "$A.1?$") at the A-interface, one input and one output (called "$E.1?$" and "$E.1!$", respectively) at the E-interface, and a single output (called "$B.1!$") at the B-interface.

The behavior of the system is then described in System 11 using the following notation. The keyword **once** signifies that the corresponding sub-routine is evaluated once the condition specified after the keyword (here $A.1? \neq \square$) becomes true; the specific statement $E.1! \leftarrow A.1?$ effects that the described value is assigned to the output variable. The keyword **upon** used for the input at sub-interface $E.1?$ can be seen as a shortcut for the statement **once** $E.1? \neq \square$ and is used for a more compact notation. (We have used both keywords in the description of the system only to explain their meaning.)

Note that multiple output variables can be set in a single such evaluation, the description of the system must assure that the value of each output variable is not changed once it is set. This is obvious in the considered case, because each output variable is set only in a single sub-routine, and each such sub-routine is evaluated only once.

Several systems will actually be specified with respect to an infinite number of inputs and are hence formally not finitary as a monotone discrete system. As all our security statements will be parametrized, each particular

System 11 Insecure channel $- \rightarrow$

```
once A.1? ≠ □
    E.1! ← A.1?
end.
upon E.1?
    B.1! ← E.1?
end.
```

parametrized resource will be a finitary system. In this sense, such resources with infinitely many inputs can be seen as a "prototype" from which the finitary objects are derived by properly restricting them.

A note on interface names of converters. In Section 3.2.3, we have described the application of a converter at interface I of a resource by connecting each sub-interface of the converter's inside interface with the sub-interface of the resource's I-interface that has the same name relative to the interface. As we label the input sub-interfaces of the discrete resources by names of the type $i?$ and the output sub-interfaces by names of the type $j!$, this naming convention must be inverted for the inside interface of the converter. The outside interface of the converter has the same name conventions as the interfaces of the resource. As the converter is directed, the composition need only be defined for connecting the inside interface of a converter to an interface of the resource, and the described naming scheme is sufficiently general.

Chapter 4

Constructing Resources for Secure Communication

Following the paradigm of constructive cryptography, the goal of a cryptographic protocol is to construct, in the sense described in Chapter 3, a desired resource from assumed resources. In the context of secure communication, the most important types of resources are communication channels with different security guarantees (described in Section 4.1) and resources formalizing shared randomness such as cryptographic keys (described in Section 4.2).

Section 4.3 discusses message authentication based on a shared uniform random function or on a weakly unforgeable MAC, and (symmetric) encryption is discussed in Section 4.4 for the particular cases of the one-time pad and CBC-mode encryption. For the one-time pad, we discuss both the case where the ciphertext is transmitted over an authenticated channel, and where it is transmitted over an insecure channel. These statements appeared in similar form, but within a slightly different formal model, in the lecture notes of Maurer [Mau14]. Furthermore, we show how several game-based notions for symmetric encryption schemes from the literature relate to constructions; in particular, we provide such relations for IND-CPA, IND-CCA, and INT-CTXT. The latter material appeared in the work of Maurer et al. [MRT12].

The construction of a secure channel via symmetric encryption and a MAC scheme can be performed in two orders: Either, one first constructs an authenticated channel using the MAC and then applies encryption to obtain a secure channel; this is usually referred to as Encrypt-then-Authenticate. Or, one first constructs a confidential channel using the encryption and then applies a MAC to obtain a secure channel; this is usually referred to as Authenticate-then-Encrypt. We discuss these two possible orders in Section 4.5; the material is

partially based the results of Maurer and Tackmann [MT10].

Section 4.6 describes how anonymous communication can be viewed as a resource. We focus on receiver-anonymous communication, i.e., a setting with one sender and multiple potential receivers, and where the identity of the intended receiver is protected. An *anonymity-preserving* public-key encryption scheme has the goal of constructing a receiver-anonymous *confidential* channel from a receiver-anonymous *insecure* channel and authenticated channels from the receivers to the sender; this extends the features of general public-key encryption as described by Coretti et al. [CMT13a]. The material of this section is taken from Kohlweiss et al. [KMO+13].

Finally, Section 4.7 discusses key establishment in the setting where only one party is authenticated. This setting prevails in practical client-server applications on the Internet where often only the server has a certified public key; the client is authenticated only later by sending its credentials over the connection that is already secured with the key. We present a protocol which is efficient and has mild computational assumptions, and also show how this protocol can be used in a practical scenario where a public-key infrastructure and a pre-shared password are available. The material is taken from Maurer et al. [MTC13].

4.1 Communication Channels

The capability of two honest parties to communicate is captured as a resource by modeling it as an explicit *communication channel* that allows the sender to input a message and the receiver to retrieve it. The security properties of such a channel are described by the capabilities provided to a third (hypothetical) type of entity: the attacker. In this section, we describe channels that model several natural types of communication such as insecure, authenticated, or secure communication, as sketched in Table 1.1 in Section 1.2. Section 4.1.1 contains material that is relevant to all channels formalized here. The subsequent sections contain the specification of insecure (Section 4.1.2), authenticated (Section 4.1.3), confidential (Section 4.1.4), and secure (Section 4.1.5) channels.

4.1.1 Formal Type and Availability Condition

All channels are formally specified as (pairs of, see Section 3.3) discrete systems in the sense of Definition 3.18, that is, they are probabilistic monotone functions from and to tuples of values. The channels are resources in a cryptographic algebra with interface labels in $\mathcal{I} = \{A, B, E\}$. In this section, we

specify all channels as channels from A to B, the attacker's capabilities are specified as interface E. The channels in the opposite direction can be described analogously, and we denote them with the "mirrored" arrow such as, e.g., $\leftarrow\!\bullet$ instead of $\bullet\!\leftarrow\ \rightarrow$.

Each channel is parametrized by a set $\mathcal{M}$, the *message space*, which describes the set of messages that can be transmitted over the channel. The message space is usually indicated by specifying it on top of the arrow symbol; we omit the set in case $\mathcal{M} := \{0,1\}^*$. The A-interface has input sub-interfaces identified by numbers $i \in \mathbb{N}$ for sending the i-th message; the channel thus takes inputs $A.1?, A.2?, \ldots$ with values in $\mathcal{M}$. The B-interface has output sub-interfaces which are also identified by numbers $i \in \mathbb{N}$; the channel thus provides outputs $B.1!, B.2!, \ldots$ which also take on values in $\mathcal{M}$. The inputs and outputs at the E-interface depend on the particular type of channel that is considered.

The system describing the channel in the availability condition is the same for all channels and is formalized as System 12. We use the symbol $\circ\!\leftarrow\ \overset{\mathcal{M}}{\twoheadrightarrow}\!\circ$ to indicate that the type of channel is arbitrary (i.e., the circles "$\circ$" are understood as wildcards that apply to cases the bullets "$\bullet$" may or may not be there).

System 12 Channel $\circ\!\leftarrow\ \overset{\mathcal{M}}{\twoheadrightarrow}\!\circ_{\perp}$ when no attacker is present

1: **once** $\forall j \leq i : A.j? \neq \square$
2: $\quad B.i! \leftarrow A.i?$
3: **end.**

The messages are processed in a fixed order (this is achieved by requiring $A.j? \neq \square$ for all $j \leq i$). The channel $\circ\!\leftarrow\ \twoheadrightarrow\!\circ_{\perp}$ is of the type described in Section 3.4.1, since additional inputs only lead to additional outputs, and the order in which the inputs are provided does not matter. These properties apply to all resource systems described below analogously.

The single-use channel $\circ\!\leftarrow\ \rightarrow\!\circ_{\perp}$ is described exactly as $\circ\!\leftarrow\ \twoheadrightarrow\!\circ_{\perp}$ but only takes a single input at the A-interface and immediately outputs the provided message at the B-interface.

4.1.2 Insecure Channels

Most communication channels existing in the real world, such as communication over the Internet or wireless networks, are insecure. An attacker that has access to the network can eavesdrop on transmitted messages and also determine the messages that are delivered to the receiver.

The resource system described as System 13 is an insecure channel with message space $\mathcal{M}$. The E-interface takes inputs and outputs at interfaces labeled with numbers $i \in \mathbb{N}$, and all inputs $E.1?, E.2?, \ldots$ and, respectively, outputs $E.1!, E.2!, \ldots$ are, like the inputs at the A- and the outputs at the B-interfaces, values in the set $\mathcal{M}$.

System 13 Insecure channel $\overset{\mathcal{M}}{-\twoheadrightarrow}$

1: **once** $\forall j \leq i : A.j? \neq \square$
2: $\quad E.i! \leftarrow A.i?$
3: **end.**

4: **once** $\forall j \leq i : E.j? \neq \square$
5: $\quad B.i! \leftarrow E.i?$
6: **end.**

All messages input by the sender are immediately leaked to the attacker and can be replaced with arbitrary other messages. The channel $-\twoheadrightarrow_{\perp} = \circ\!- \twoheadrightarrow\!\circ_{\perp}$ is obtained by attaching at the E-interface the converter that simply forwards all messages to the receiver. The existence of such a converter is necessary for the resource $-\twoheadrightarrow$ to be well-defined as described in Section 3.3. The single-message channel that takes only a single input $A.1?$ at the A-interface, provides a single output $B.1!$ at the B-interface, and takes one input $E.1?$ and provides one output $E.1!$ at the E-interface is denoted as $-\rightarrow$, and is described as an example in System 11 on page 97.

4.1.3 Authenticated Channels

An authenticated channel guarantees that all messages delivered to the receiver B have actually been sent by the designated sender A. Such channels may exist physically (if one assumes, e.g., that the receiver can verify the sender's voice or handwriting). They can also be constructed by cryptographic schemes such as signatures or MAC. There are different types of authenticated channels that differ in their guarantees in terms of whether, e.g., the order of the messages is preserved, or whether messages sent honestly once can be delivered repeatedly; we describe two specific but natural types below.

Unordered authenticated channel. The channel $\bullet\!-\!\diamond\!-\!\twoheadrightarrow$ described as System 14 models the authenticated channel with message space $\mathcal{M}$ which only guarantees, for each message output at the B-interface, that it has been input at the A-interface before. This means that the attacker has the capabilities of delivering the messages in a different order, delivering the same message

multiple times, and to never deliver certain messages. This behavior is indicated by the "$\diamond$" used in the symbol, which indicates a buffer that stores the values input by A. The messages stored in the buffer can later be delivered to B.

The E-interface again has input and output sub-interfaces with labels $i \in \mathbb{N}$. The outputs $E.1!, E.2!, \ldots$ are values in the set $\mathcal{M}$; the channel outputs at each sub-interface $E.i!$ the message input at the $A.1?$-interface. The inputs $E.1?, E.2?, \ldots$ are values in the set $\mathbb{N}$. Upon input $E.i? = j$, the channel outputs as the i-th message at the B-interface the j-th message that was input at the A-interface, more formally, $B.i! \leftarrow A.j?$. It is again easy to verify that

System 14 Unordered authenticated channel $\bullet\!\!-\!\!\overset{\mathcal{M}}{\diamond}\!\!-\!\!\twoheadrightarrow$

```
1: once ∀j ≤ i : A.j? ≠ □
2:     E.i! ← A.i?
3: end.

4: once ∀j ≤ i : B.j! ≠ □
5:     k ← E.i?
6:     once A.k? ≠ □
7:         B.i! ← A.k?
8:     end.
9: end.
```

the channel is a monotone system of the type described in Section 3.4.1, and that there is a converter that can be attached to the E-interface to obtain the resource system $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow_{\perp} = \circ\!\!- \; -\!\!\twoheadrightarrow\!\!\circ_{\perp}$. This converter, upon receiving an output at $E.i!$, immediately inputs $E.i? \leftarrow i$.

Ordered authenticated channel. The channel $\bullet\!\!- \; -\!\!\twoheadrightarrow$ as specified in System 15 formalizes the stronger guarantee that the messages input at the A-interface are output at the B-interface in the same order, and that no messages can be dropped in between. This guarantee is useful in applications because the channel can be understood as faithfully delivering a "stream" of messages.

The E-interface again has input and output sub-interfaces with labels $i \in \mathbb{N}$. The outputs $E.1!, E.2!, \ldots$ are again values in the set $\mathcal{M}$; the channel immediately outputs the messages input at the corresponding A-sub-interface. The inputs $E.1?, E.2?, \ldots$ are unary and only take a special value $\uparrow$ (the value space is $\{\uparrow\}$); the understanding is that as soon as the inputs $A.i?$ and $E.i?$ are provided, the variable $B.i!$ is defined to be equal to $A.i?$.

The single-use authenticated channel is denoted as $\bullet\!\!- \; -\!\!\rightarrow$ and behaves as $\bullet\!\!- \; -\!\!\twoheadrightarrow$. Analogously to $- \; -\!\!\rightarrow$, the A-interface has a single input $A.1?$, the B-interface has a single output $B.1?$, and the E-interface has one input $E.1?$

System 15 Ordered authenticated channel $\bullet\!\overset{\mathcal{M}}{\leftarrow}\!\twoheadrightarrow$

1: **once** $\forall j \leq i : A.j? \neq \square$
2: $\quad E.i! \leftarrow A.i?$
3: **end.**

4: **once** $\forall j \leq i : E.j? = \uparrow \wedge A.j? \neq \square$
5: $\quad B.i! \leftarrow A.i?$
6: **end.**

and one output $E.1!$.

4.1.4 Confidential Channels

A confidential channel protects the secrecy of the transmitted messages, but does not necessarily also guarantee the integrity or authenticity. Confidentiality means that at most some partial information (usually the length) about the transmitted messages is output at the E-interface. The absence of integrity and authenticity guarantees means that E-interface may allow to modify the messages input by A before they are output at B, or to inject messages into the channel which are delivered to B. We describe here two specific types of confidential channels which will be of interest in subsequent sections to describe what certain types of schemes achieve.

Non-malleable confidential channel. The first such channel is a *non-malleable* confidential channel which formalizes that the E-interface allows to either forward (unmodified) messages that have been input at the A-interface, or to input "unrelated" messages that will then be output at the B-interface. Such a channel from A to B can, for instance, be constructed (from an authenticated channel from B to A and an insecure channel from A to B) by a public-key encryption scheme, as shown by Coretti et al. [CMT13a].

The channel, which is denoted by the symbol $-\!\diamond\!\twoheadrightarrow\!\bullet$, is specified as System 16. The outputs $E.1!, E.2!, \ldots$ at the E-interface are values in the set $\mathbb{N}$, each output $E.i!$ is defined as the length of the corresponding input provided at $A.i?$. The inputs $E.1?, E.2?, \ldots$ are in the set $\mathcal{M} \cup \mathbb{N}$. The behavior is that if an input $E.i?$ has a value $m \in \mathcal{M}$, then the output $B.i!$ is set to this value m. If the input $E.i?$ has a value $k \in \mathbb{N}$, then the output $B.i!$ is set to the value $A.k?$, that is, the k-th input at the A-interface is provided as i-th output at the B-interface. The "$\diamond$" in the symbol again signifies that the channel contains a "buffer" such that the messages input by A can be output at the B-interface in any order and each message can even be output multiple times.

System 16 Non-malleable confidential multiple-use channel $\overset{\mathcal{M}}{-\diamond\twoheadrightarrow\bullet}$

```
once ∀j ≤ i : A.j? ≠ □
    E.i! ← |A.i?|
end.

once (∀j < i : B.j! ≠ □) ∧ E.i? ≠ □
    if E.i? ∈ ℕ then
        k ← E.i?
        once A.k? ≠ □
            B.i! ← A.k?
        end.
    else                                ▷ E.i? ∈ M
        B.i! ← E.i?
    end if
end.
```

XOR-malleable confidential channel. An exemplary channel that provides at the E-interface the capability to modify messages input by A but not to input unrelated messages is described as System 17. We refer to this channel as the *XOR-malleable* channel because the channel takes at the E-interface an "XOR-mask" that is applied to the message input at interface A. For simplicity, we describe the channel only for a single message and with message space $\mathcal{M} = \{0,1\}^\ell$. This channel can be constructed by applying the one-time pad to an insecure communication channel and a shared secret key. We describe this construction (but for messages of arbitrary length) in Theorem 4.5 in Section 4.4.1.

System 17 XOR-malleable confidential single-use channel $\overset{\{0,1\}^\ell}{-\oplus\rightarrow\bullet}$ for ℓ-bit messages

```
upon A.1?
    if E.1? ≠ □ then
        E.1! ← A.1? ⊕ E.1? ⊕ B.1!
    else
        E.1! ←$ {0,1}^ℓ
    end if
end.

upon E.1?
    if A.1? ≠ □ then
        B.1! ← A.1? ⊕ E.1? ⊕ E.1!
    else
        B.1! ←$ {0,1}^ℓ
    end if
end.
```

Both the input $E.1?$ and the output $E.1!$ at the E-interface are in the set $\{0,1\}^\ell$. In a less formal description, the channel behaves as follows:

- If first a message m is input at the A-interface, then the channel outputs

at the E-interface a uniformly random bit string of length ℓ. If later a message is input at the E-interface, then the channel outputs at the B-interface the XOR of the original message m and both the output and input at the E-interface.

- If first a message c' is input at the E-interface, then the channel outputs at the B-interface a uniformly random bit string m' of length ℓ. If later a message m is input at the A-interface, then the channel outputs at the E-interface the value $m \oplus m' \oplus c'$.

Note that although the description of the channel differs for the case where input at $A.1?$ is provided before input at $E.1?$ is provided and the case where the inputs appear in the opposite order, the channel is still a monotone system in the sense of Section 3.4.1. The output at the E-interface is a uniformly distributed string of ℓ bits, and the output at the B-interface equals the XOR of the input at the A-interface and the input and output at the E-interface.

The converter which is necessary to assure that the pair of $\overset{\{0,1\}^\ell}{\multimap\!\!\oplus\!\!\to}\bullet$ and $\overset{\{0,1\}^\ell}{\multimap\!\!\oplus\!\!\to}\bullet_\perp = \circ\!\!- \to\!\circ_\perp$ is a well-defined resource is again easy to describe: Upon input a bit string $c \in \{0,1\}^\ell$ at interface $\mathsf{In}.1!$, output $\mathsf{In}.1? \leftarrow c$.

4.1.5 Secure Channels

A secure channel protects both the authenticity and the confidentiality of the transmitted messages. Similar discussions as in the cases of authenticated channels in Section 4.1.3 and about confidentiality in Section 4.1.4 also apply here. Analogously to the case of authenticated channels, we again describe an ordered and an unordered version of the channel; and analogously to the case of confidential channels we consider channels that leak only the message length.

Unordered secure channel. The unordered secure communication channel $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\!\bullet$ described as System 18 formalizes the guarantee that the messages are transmitted confidentially and authentically, but the attacker at the E-interface can deliver the messages in an arbitrary order, and each message can potentially be delivered multiple times. The inputs and outputs at interface E are again labeled by numbers $i \in \mathbb{N}$. The outputs $E.1!, E.2!, \ldots$ are values in $\mathbb{N}$ and are set to the lengths of the corresponding messages input at interface A. The inputs $E.1?, E.2?, \ldots$ are values in $\mathbb{N}$ and specify, as for $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$, which message shall be output at the B-interface.

System 18 Unordered secure channel $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\!\bullet$

```
1: once ∀j ≤ i : A.j? ≠ □
2:     E.i! ← |A.i?|
3: end.

4: once ∀j ≤ i : B.j! ≠ □
5:     k ← E.i?
6:     once A.k? ≠ □
7:         B.i! ← A.k?
8:     end.
9: end.
```

Consistently with the previously described channels, there is again a converter that, upon receiving an output at $E.i!$ of $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\!\bullet$, sets the input $E.i? \leftarrow i$ to achieve the behavior of $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\!\bullet_{\perp} = \circ\!\!- \twoheadrightarrow\!\!\circ_{\perp}$.

Ordered secure channel. The secure communication channel $\bullet\!\!- \twoheadrightarrow\!\!\bullet$ described as System 19 formalizes the guarantee that the messages input at the A-interface are output at the B-interface in the same order, and no messages can be dropped in between. Interface E outputs the lengths of the messages input at the A-interface, and takes as input unary messages which trigger the output of the respective messages at the B-interface, which models that the attacker can always suppress the communication from A to B by not providing these inputs. Such a channel is useful in applications because it guarantees that a stream of messages is transmitted securely from A to B, the only capability of the attacker is to interrupt the communication.

Roughly, this channel can be seen as the goal of protocols like TLS and ssh. The case of IPsec is slightly different, because the protocol does not guarantee that all messages (in a sequence) are delivered [Jos14].

System 19 Ordered secure channel $\bullet\!\!- \twoheadrightarrow\!\!\bullet$

```
1: once ∀j ≤ i : A.j? ≠ □
2:     E.i! ← |A.i?|
3: end.

4: once ∀j ≤ i : E.j? = ↑∧A.j? ≠ □
5:     B.i! ← A.i?
6: end.
```

The outputs $E.1!, E.2!, \ldots$ are values in the set $\mathbb{N}$; the channel outputs the length of the respective messages input at the A-interface. The inputs $E.1?, E.2?, \ldots$ are unary and only take a value $\uparrow$; the understanding is that as soon as the inputs $A.i?$ and $E.i?$ are provided, the variable $B.i!$ is defined to be equal to $A.i?$. The single-use channel $\bullet\!\!- \rightarrow\!\!\bullet$ is again obtained by restricting System 19 to inputs $A.1?$ and $E.1?$ and outputs $B.1!$ and $E.1!$.

4.1.6 Parametrized Channels

To formalize construction statements in which the advantage of the distinguisher is parametrized depending on the assumed "use" of the scheme, such as the number of communicated bits or messages, we describe in this section families of monotone binary outputs that formalize such a restriction and are useful to make parametrized construction statements as described in Section 3.4.2.

Number of communicated bits. The number of communicated bits is formalized by an MBO that becomes 1 once more than ℓ bits have been:

- input at interface A, or
- output at interface B, or
- input at interface E, or
- output at interface E.

A channel parametrized via this MBO is written as $\circ\!\!-\!\overset{\ell\text{ bits}}{\twoheadrightarrow}\!\circ$. The MBO for converters is defined analogously: Once ℓ bits have been input or output at the outside interface, the MBO becomes 1.

Number of communicated messages. A different type of parametrization considers the number of communicated messages. This is also formalized by an MBO that becomes 1 once more than q messages have been input or output at any interface (analogously to the MBO above). A channel parametrized via this MBO is written as $\circ\!\!-\!\overset{q\text{ msgs}}{\twoheadrightarrow}\!\circ$. The MBO for converters is defined analogously: Once q messages have been input or output at the outside interface, the MBO becomes 1.

4.2 Keys and Shared-Randomness Resources

An important type of resource for cryptographic protocols is shared randomness. This can be a cryptographic key (a shared random bit string) or "interactive" objects like a shared random function. Cryptographic schemes that make use of shared randomness are often referred to as "symmetric" schemes such as symmetric encryption schemes or message authentication codes, as opposed to "asymmetric" public-key schemes where the key material used by the two involved parties is different.

4.2.1 General Shared-Randomness Resources

For a random system $\mathbf{R}$, we define the system $\overleftrightarrow{\mathbf{R}}$ as the resource with interfaces A, B, and E, where access to the same instance of the random system is provided to both A and B. The E-interface provides the capability to interrupt the access of the parties to the resource, but does not allow to further access the contained random system.

As an example, consider a uniform random function (URF) $F_n : \{0,1\}^* \to \{0,1\}^n$, which can be viewed as a random system $\mathbf{F}_n$ taking a sequence of inputs $X_1, X_2, \ldots \in \{0,1\}^*$ and, for each input X_i, providing an output $Y_i \in \{0,1\}^n$. The URF, as a random system, responds to an input X_i with uniform random outputs $Y_i \in_R \{0,1\}^n$, with the restriction that if two inputs $X_i = X_j$ match, then the outputs $Y_i = Y_j$ also match.

The shared uniform random function is described as a resource $\overleftrightarrow{\mathbf{F}}_n$ in Systems 20 and 21. The resource takes at the A- and B-interfaces inputs $A.1?, A.2?, \ldots$ and $B.1?, B.2?, \ldots$ with values in $\{0,1\}^*$, and provides outputs $A.1!, A.2!, \ldots$ and $B.1!, B.2!, \ldots$ with values in $\{0,1\}^n$.

System 20 Shared URF without attacker $\overleftrightarrow{\mathbf{F}}_{n\perp}$

Choose instance $F_n : \{0,1\}^* \to \{0,1\}^n$

once $\forall j \leq i : A.j? \neq \square$
    $A.i! \leftarrow F_n(A.i?)$
end.

once $\forall j \leq i : B.j? \neq \square$
    $B.i! \leftarrow F_n(B.i?)$
end.

At the E-interface, $\overleftrightarrow{\mathbf{F}}_n$ takes "trigger" inputs to "activate" the interfaces A and B. These inputs $E.A?$ and $E.B?$ are unary and only take the fixed value "$\uparrow$." The trigger input at the E-interface allows to take into account that shared-randomness resources are usually constructed from cryptographic keys, and these are in turn constructed via a key-establishment protocol which can be delayed or interrupted by the attacker.

4.2.2 Shared Secret Keys

As a particular instance of shared-randomness resources, we formalize cryptographic keys. Following the above approach, we denote the random system that takes as input a trigger (denoted "$\uparrow$") and responds with a uniform κ-bit

System 21 Shared URF $\overleftrightarrow{\mathbf{F}}_n$

```
Choose instance F_n : {0,1}^* → {0,1}^n

once (E.A? ≠ □) ∧ (∀j ≤ i : A.j? ≠ □)
    A.i! ← F_n(A.i?)
end.

once (E.B? ≠ □) ∧ (∀j ≤ i : B.j? ≠ □)
    B.i! ← F_n(B.i?)
end.
```

string as $\mathbf{U}_\kappa$ and regard the shared random κ-bit string $\overleftrightarrow{\mathbf{U}}_\kappa$ as being a cryptographic key, which simply outputs a random key at both interfaces A and B. Note that the random system $\mathbf{U}_\kappa$ takes only a single input, it is hence fully described by the conditional distribution $\mathsf{p}^{\mathbf{U}_\kappa}_{Y_1|X_1}$.

We denote the described resource by a special symbol $\bullet\!\!=\!\!=\!\!\bullet \coloneqq \overleftrightarrow{\mathbf{U}}_\kappa$ and describe it formally as Systems 22 and 23. This system models the shared secret key that is assumed by (symmetric) encryption and authentication schemes and could result from a key establishment protocol.

Let $\mathcal{K}$ be a discrete set, the *key space*. (Usually, we consider $\mathcal{K} \coloneqq \{0,1\}^\kappa$ for some $\kappa \in \mathbb{N}$.) A key with key space $\mathcal{K}$ is a resource that draws a key $k \in \mathcal{K}$ uniformly at random and outputs it to both A and B. In particular, both interfaces only have a single unary input $A.1?$ and $B.1?$ and a single output $A.1!$ and $B.1!$ which takes values in $\mathcal{K}$.

System 22 Key without attacker $\circ\!\!=\overset{\mathcal{K}}{=}\!\!\circ_\perp$

```
k ←$ K

upon A.1?
    A.1! ← k
end.

upon B.1?
    B.1! ← k
end.
```

The E-interface takes "trigger" inputs to "activate" the interfaces A and B. These inputs $E.A?$ and $E.B?$ are unary and only take the fixed value "$\uparrow$." The symbol "$\circ\!\!=\!\!=\!\!\circ$" with the circles "$\circ$" instead of the bullets "$\bullet$" again indicates that the behavior is the same for keys where the parties are authenticated and keys where the parties are not authenticated.

System 23 Key $\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet$

1: $k \leftarrow_{\$} \mathcal{K}$
2: **once** $(A.1? \neq \square) \wedge (E.A? \neq \square)$
3: $\quad A.1! \leftarrow k$
4: **end.**
5: **once** $(B.1? \neq \square) \wedge (E.B? \neq \square)$
6: $\quad B.1! \leftarrow k$
7: **end.**

Unilaterally authenticated keys. A shared key is said to be *unilaterally authenticated* or *unilateral* if only one of the two parties has the guarantee that the key is indeed shared with the intended partner, the other party has no such guarantee. We describe this guarantee as a resource $=\!=\!\bullet$ that is more formally specified as System 24. In case no attacker is present, the resource simply outputs a uniform random key to both A and B, which is exactly the same behavior as specified in System 22.

The A and B-interfaces of the resource $=\!=\!\bullet$ are similar to the mutually authenticated case. The E-interface has three inputs $E.A?$, $E.B?$, and $E.\texttt{key}?$. The input $E.B?$ is binary (i.e., takes a value in $\{0,1\}$) and determines that either (in case $E.B? = 0$) the server shares a secret key with the client, or (in case $E.B? = 1$) the server shares a key with the attacker. In the first case, i.e. $E.B? = 0$, a uniformly random key is output at interface $B.1!$, and upon input a unary "trigger" message at $E.A?$, the same key is also output at interface $A.1!$. In the second case, i.e. $E.B? = 1$, upon receiving a value $k \in \mathcal{K}$ as input $E.\texttt{key}?$, this value k is output at interface $B.1!$.

The unilateral key captures what is achieved by a key establishment protocol where only one of the parties is authenticated. This scenario is prevailing in the Internet, where most protocols have a client-server structure and only the server has a certified public key.

4.2.3 Parametrized Randomness Resources

The shared-randomness resources like uniform random functions and uniform random permutations can also be parametrized. The typical parametrization is in terms of the number of evaluations via the A and B-interfaces. This parametrization is formalized by an MBO that becomes 1 once more than q queries have been made at interface A, or more than q queries have been made at interface B. A resource $\mathbf{R}$ parametrized by this MBO is written as $\mathbf{R}^{|q}$. The MBO for converters is defined analogously: Once q queries have been made at the outside interface, the MBO becomes 1.

System 24 Unilateral key $= \!=\!\bullet$

$k \leftarrow_{\$} \mathcal{K}$

once $(B.1? \neq \square \neq E.\texttt{key}?) \wedge (E.B? = 1)$
  $B.1! \leftarrow E.\texttt{key}?$
end.

once $(B.1? \neq \square) \wedge (E.B? = 0)$
  $B.1! \leftarrow k$
end.

once $(A.1?) \wedge (E.B? = 0) \wedge (E.A? \neq \square)$
  $A.1! \leftarrow k$
end.

4.3 Message Authentication

Message authentication can be achieved based on a shared randomness resource. In this section, we show two constructions. The first construction is based on a shared uniform random function and is implemented by simply evaluating the random function on the message and appending the output to the original message. As a formula, the achieved construction can be written as

$$\left(\overleftrightarrow{\mathbf{F}}, - \twoheadrightarrow\right) \quad \xrightarrow{\textsf{aut}} \quad \bullet\!-\!\diamond\!-\!\twoheadrightarrow,$$

where $\overleftrightarrow{\mathbf{F}}$ denotes the shared URF, $- \twoheadrightarrow$ a multi-message insecure channel, and $\bullet\!-\!\diamond\!-\!\twoheadrightarrow$ an unordered authenticated channel as described in Section 4.1.3. Here, we denote by **aut** the scheme where the sender appends the output of the URF to the message, and the receiver checks whether the received value equals the one obtained by also evaluating the URF on the message.

We also show that a weakly unforgeable MAC scheme achieves a similar construction. Indeed, for a protocol **mac** based on a MAC function (but otherwise similar to **aut**) the following construction is achieved:

$$(\bullet\!=\!=\!\bullet, - \twoheadrightarrow) \quad \xrightarrow{\textsf{mac}} \quad \bullet\!-\!\diamond\!-\!\twoheadrightarrow.$$

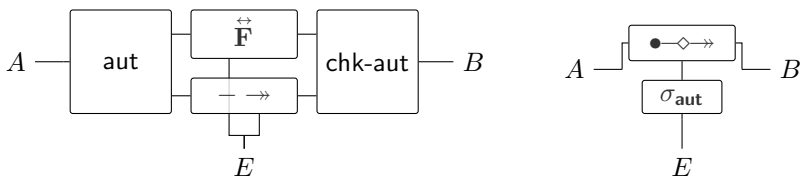
The protocol **mac** computes the MAC function using the message and the shared secret key and appends the obtained tag to the message. The receiver, for a received pair of message and tag, checks whether the tag matches the message with respect to the shared key. The security proof is a reduction from

the WUF-CMA property of the MAC scheme.

4.3.1 Construction Based on a Shared URF

In a message-authentication protocol that uses a shared uniform random function (URF), the sender can simply evaluate the URF on a given message m and send the message m together with the authentication tag t obtained by the URF. The receiver, who also has access to the URF, can also evaluate the URF on a received message m' and compare the output t' of the URF to the received authentication tag. He will accept the received message only if the received and the computed tags match. The intuition behind the protocol is that an attacker, to inject or modify a message such that it is accepted by the receiver, would have to predict the output of the URF. Theorem 4.2 proven below confirms that this intuition is correct and the protocol is indeed secure.

Let $\mathcal{M} := \{0,1\}^*$ be the message space and consider a shared URF $F_n : \mathcal{M} \to \{0,1\}^n$ as described in Section 4.2.1. The simple protocol described above, which we formalize as a pair **aut** $= (\mathsf{aut}, \mathsf{chk\text{-}aut})$ of converters below, constructs an authenticated channel $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$ (with message space $\mathcal{M}$) from a shared URF $\overleftrightarrow{\mathbf{F}}$ and an insecure channel $-\!\!-\!\!\twoheadrightarrow$ (with message space $\mathcal{M}$). The two settings that occur in the security statement are depicted in Figure 4.1.



(a) The authentication protocol **aut** $= (\mathsf{aut}, \mathsf{chk\text{-}aut})$ applied to the shared URF and the insecure channel.

(b) The authenticated channel $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$ with a simulator $\sigma_{\mathbf{aut}}$ at the E-interface.

Figure 4.1: The two settings involved in the of construction statement about the authentication protocol.

The two converters aut and chk-aut are described in more detail as follows, with the understanding that they process the incoming inputs (or messages) in order.

aut: Upon input a message $m \leftarrow \mathsf{Out}.i? \in \mathcal{M}$, query $\mathsf{In}.1.i? \leftarrow m$ at the first inside sub-interface,[1] obtaining a value $t \leftarrow \mathsf{In}.1.i! \in \{0,1\}^n$. Output the

[1]Recall that at the inside interface of a converter, the interface labels for input and output are

concatenation $m.t$ at the second inside sub-interface, i.e., $\mathsf{In}.2.i? \leftarrow m.t$.

chk-aut: Upon input a value $x \leftarrow \mathsf{In}.2.i! \in \{0,1\}^*$ at the second inside sub-interface, if $|x| < n$ then halt. Otherwise, parse $m'.t' \leftarrow x$ (this is unique, because $t' \in \{0,1\}^n$) and query $\mathsf{In}.1.i? \leftarrow m'$ at the first inside sub-interface, obtaining $t'' \leftarrow \mathsf{In}.1.i! \in \{0,1\}^n$. If $t' = t''$, then output $\mathsf{Out}.i! \leftarrow m'$ at the outside interface.

The message flow of the MAC protocol applied to the resources $\overleftrightarrow{\mathbf{F}}_n$ and $- \twoheadrightarrow$ is depicted in Figure 4.2. The interface names used in the descriptions of the systems are used to label the arrows. Since the shared URF $\overleftrightarrow{\mathbf{F}}_n$ has the label 1 in the tuple, and the insecure channel $- \twoheadrightarrow$ has the label 2, the interfaces connect as shown in the figure.

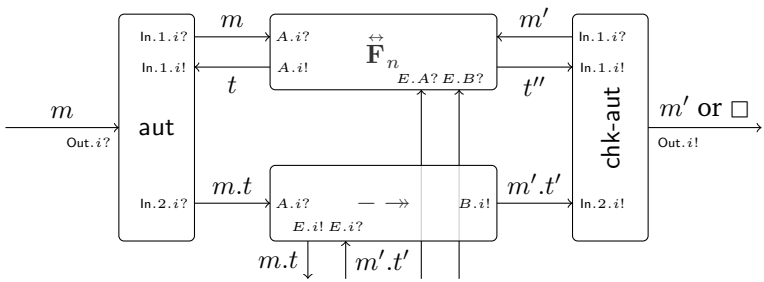


Figure 4.2: The message flow in the authentication protocol.

The construction is proven relative to a simulator σ_{aut} which behaves as follows. It internally keeps a URF[2] $R : \mathcal{M} \to \{0,1\}^n$ and, upon receiving an input $m_i \leftarrow \mathsf{In}.i! \in \mathcal{M}$ at the inside interface (i.e., from the authenticated channel $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$), and once $\mathsf{Out}.1.A? = \uparrow$,[3] output $\mathsf{Out}.2.i! \leftarrow m_i.R(m_i)$ at the outside interface. Upon receiving in input $m'.t' \leftarrow \mathsf{Out}.2.j?$ at the outside interface, once the input $\mathsf{Out}.1.B? = \uparrow$ has been provided at the outside interface and $m'.t' = m_i.t_i$ for some $i \in \mathbb{N}$, then output $\mathsf{In}.j? \leftarrow i$ at the inside interface.

The composition of the simulator σ_{aut} and the authenticated channel $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$ is depicted in Figure 4.3, which shows how the two systems connect and which messages are exchanged. The interface labels are indicated for each message input or output by a system. The simulator is a discrete monotone

reversed, as described on page 97.

[2]This can be done efficiently by sampling the URF adaptively.

[3]That sub-interface corresponds to the A-sub-interface of the E-interface of $\overleftrightarrow{\mathbf{F}}$, hence the trigger means that A can now use the URF.

system in the sense of Definition 3.18 because each output at the outside interface only depends on the corresponding input at the inside interface, and an input at the inside interface is given once the corresponding output has been provided at the outside interface and the message is in the buffer of the channel.

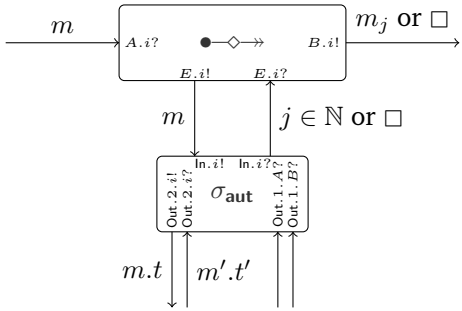


Figure 4.3: The message flow in the execution with the simulator.

The proof of the following theorem formalizes that, to break the security of the scheme, the attacker would have to predict an output of the URF. Since the values of the URF are uniformly random, the probability of succeeding in q queries is bounded by $\frac{q}{2^n}$, where n is the output length of the URF. The bound we prove is larger than expected by a factor of 2. The reason lies in the simulator that we describe; it is a monotone discrete system. In a generalization of the framework where non-monotone systems are allowed, one can prove a bound of $\frac{q}{2^n}$ (instead of $\frac{q}{2^{n-1}}$).

Theorem 4.1. *Let* **aut** $= (\mathsf{aut}, \mathsf{chk\text{-}aut})$ *be the protocol described above. For the simulator* σ_{aut}*,*

$$\left(\overleftrightarrow{\mathbf{F}}_n, - \twoheadrightarrow\right) \xrightarrow{\mathsf{aut}, \sigma_{\mathsf{aut}}, q/2^{n-1}} \bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow,$$

where the statement is parametrized in $q \in \mathbb{N}$ *for the resources* $\overleftrightarrow{\mathbf{F}}_n^{|q}$, $\overset{q \text{ msgs}}{-\twoheadrightarrow}$*, and* $\overset{q \text{ msgs}}{\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow}$.

Proof. For the availability condition, we show that

$$\mathsf{aut}^A \mathsf{chk\text{-}aut}^B \left(\left(\overleftrightarrow{\mathbf{F}}_n\right)^{|q}_{\perp}, \overset{q \text{ msgs}}{-\twoheadrightarrow}_{\perp} \right) \quad \equiv \quad \overset{q \text{ msgs}}{\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow}_{\perp}.$$

This follows directly from the correctness of the scheme: for a message $m \in \mathcal{M}$, the shared URF responds with the same tag $t \in \{0,1\}^n$ at the A- and B-interfaces, hence $\mathsf{chk\text{-}aut}$ accepts all tags generated by aut. This means that in

both cases an input of a message m at the A-interface will lead to an output of the same message at the B-interface. In both cases, the system provides output until q messages have been input at the A-interface.

For the security condition, we define

$$\mathbf{R}_q \;\coloneqq\; \mathsf{aut}^A\mathsf{chk\text{-}aut}^B\left(\overleftrightarrow{\mathbf{F}}_n^{|q}, \overset{q\text{ msgs}}{-\!\!\rightarrow}\right) \qquad \text{and} \qquad \mathbf{S}_q \;\coloneqq\; \sigma_{\mathsf{aut}}{}^E\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow^{q\text{ msgs}}.$$

The systems $\mathbf{R}_q$ and $\mathbf{S}_q$ are random systems which input and output tuples in the set $\mathcal{X}^{(\mathcal{J})}$ (for $\mathcal{J} = \mathcal{I} \times \mathbf{\Lambda}$). We define an MBO $A_1, A_2, \ldots$ that becomes 1 as soon as an input $(\tilde{m}, \tilde{t}) \in \mathcal{M} \times \{0,1\}^n$ is provided at the E-interface such that $\tilde{m}$ was not input at the A-interface before, but $R(\tilde{m}) = \tilde{t}$ according to the URF $R : \mathcal{M} \to \{0,1\}^n$ within $\overleftrightarrow{\mathbf{F}}_n$ (for $\mathbf{R}_q$) or σ_{aut} (for $\mathbf{S}_q$), respectively.

Both $\hat{\mathbf{R}}_q$ and $\hat{\mathbf{S}}_q$ are conditionally equivalent to a system $\mathbf{H}$ that behaves like $\mathbf{R}_q$ but whenever a message m is input at the A-interface such that previously pairs $m.t_1, \ldots, m.t_\ell$ have been input at the E-interface, the system outputs a pair $m.t$ with $t \in_R \{0,1\}^n \setminus \{t_1, \ldots, t_\ell\}$. As this is the same distribution that is also generated by $\mathbf{R}_q$ given that the MBO is 0 (since the output of the URF, given that it is not in the set $\{t_1, \ldots, t_n\}$, is uniform over the remaining values. This means that $\hat{\mathbf{R}}_q \;\models\; \mathbf{H}$. Analogously it holds that $\hat{\mathbf{S}}_q \models \mathbf{H}$, since if none of the "guesses" is correct, the values output at the B-interface are exactly the messages for which the attacker used a tag that was output at the E-interface before, and system $\mathbf{S}_q$ behaves exactly like $\mathbf{H}$. By Theorem 2.16, this implies that the distinguishing advantages $[\![(\mathbf{R}_q \mid \mathbf{H})]\!]$ and $[\![(\mathbf{S}_q \mid \mathbf{H})]\!]$ can each be bounded by the probability of non-adaptively provoking the respective MBO.

As provoking the MBO corresponds to guessing the output of an n-bit uniform random function, each of these probabilities can be bounded by the probability of (non-adaptively) guessing a uniformly random n-bit string in q attempts, which is $\frac{q}{2^n}$. Overall, we obtain

$$[\![(\mathbf{R}_q \mid \mathbf{S}_q)]\!] \;\leq\; [\![(\mathbf{R}_q \mid \mathbf{H})]\!] + [\![(\mathbf{H} \mid \mathbf{S}_q)]\!] \;\leq\; \left[\!\!\left[\hat{\mathbf{R}}_q\right]\!\!\right] + \left[\!\!\left[\hat{\mathbf{S}}_q\right]\!\!\right] \;\leq\; \frac{q}{2^n} + \frac{q}{2^n},$$

which concludes the proof. □

The proof extends to URFs which are defined on some set $\mathcal{M} \subseteq \{0,1\}^*$. The scheme then constructs an authenticated channel with message space $\mathcal{M}$ from the shared URF and an insecure channel which has message space $\mathcal{M}'$ such that $m.t \in \mathcal{M}'$ for all $m \in \mathcal{M}$ and $t \in \{0,1\}^n$.

The shared URF can be constructed from a shared secret key such as $\bullet\!\!=\;=\!\!\bullet$ by applying a pseudo-random function at the A- and B-interfaces. This can

be phrased as an independent construction step which achieves

$$\bullet\!= =\!\bullet \quad \xrightarrow{\textsf{prf}} \quad \overset{\leftrightarrow}{\mathbf{F}}_n$$

and proven for any pseudo-random function with outputs in $\{0,1\}^n$ under the corresponding assumption, where **prf** is the protocol in which A and B simply evaluate the PRF. The composition theorem then implies that

$$(\bullet\!= =\!\bullet, - \twoheadrightarrow) \quad \xrightarrow{\textsf{aut}\circ\langle\textsf{prf}\rangle_1} \quad \bullet\!-\!\diamond\!-\!\twoheadrightarrow.$$

4.3.2 Construction Based on Weakly Unforgeable MACs

The (unordered) authenticated channel can alternatively be constructed from an insecure channel and a shared secret key by a protocol based on a weakly unforgeable MAC scheme. Similarly to the protocol in Section 4.3.1, the sender evaluates the MAC function on the message and the shared key, and appends the computed tag to the message. The receiver parses the received bit string as a pair of message and tag and uses the check function defined by the MAC scheme. (For most MAC schemes, this is defined as computing the tag for the received message and comparing the computed tag to the received one.) We describe the scheme as explicitly sending the pair of message and tag; if both values are bit strings and the tag has a fixed length, the pair can simply be encoded by concatenating the two strings as in Section 4.3.1.

Let $\mathrm{MAC} = (\mathrm{mac}, \mathrm{check})$ be a MAC scheme with key space $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$ as defined in Section 2.4.2. Each such MAC scheme gives rise to a protocol **mac** $= (\mathsf{tag}, \mathsf{chk})$ as follows, where the messages are processed in order (as in the protocol in Section 4.3.1).

tag: Initially, output $\mathsf{In}.1.1?$ at the inside interface (to retrieve the key). Upon input a message $m \leftarrow \mathsf{Out}.i? \in \mathcal{M}$ at the outside interface (and once the key $k \in \mathcal{K}$ is obtained at the inside interface $\mathsf{In}.1.1!$), compute $t \leftarrow \mathrm{mac}\,(k, m)$ and output the pair (m, t) at the inside interface $\mathsf{In}.2.i?$.

chk: Initially, output $\mathsf{In}.1.1?$ at the inside interface (to retrieve the key). Upon input a pair $(m', t') \leftarrow \mathsf{In}.2.i! \in \mathcal{M} \times \mathcal{T}$ at the inside interface (and once the key $k \in \mathcal{K}$ is obtained at the inside interface $\mathsf{In}.1.1!$), if

$$\mathrm{check}\,(k, m', t') = \underline{\mathsf{true}},$$

then output $\mathsf{Out}.i! \leftarrow m'$ at the outside interface.

The construction is proven relative to a simulator $\sigma_{\textsf{mac}}$ which initially samples a key $k \leftarrow_{\$} \mathcal{K}$ and initializes a buffer $\mathcal{B} \leftarrow \emptyset$ and then behaves as follows:

- Upon receiving the i-th message $m \leftarrow \mathsf{In}.i! \in \mathcal{M}$ at the inside interface, and once the input $\uparrow$ has been provided at the outside sub-interface $\mathsf{Out}.1.A?$, compute $t \leftarrow \mathrm{mac}(k, m)$, set $\mathcal{B} \leftarrow \mathcal{B} \cup \{(i, m)\}$ and output (m, t) at the outside interface $\mathsf{Out}.2.i!$.

- Upon receiving input $(\tilde{m}, \tilde{t}) \leftarrow \mathsf{Out}.2.i? \in \mathcal{M} \times \mathcal{T}$ at the outside interface, and once the input $\uparrow$ has been provided at the outside sub-interface $\mathsf{Out}.1.B?$, if $\mathrm{check}(k, \tilde{m}, \tilde{t}) = \underline{\mathsf{true}}$ and once there is an $j \in \mathbb{N}$ with $(j, \tilde{m}) \in \mathcal{B}$, then output j at the inside interface $\mathsf{In}.i?$.

The construction is proven based on the assumption that the MAC scheme MAC is weakly unforgeable. In the proof of the theorem, we describe an explicit reduction system $\mathbf{C}_{\mathsf{mac}}$ that translates a distinguisher for the distinction problem from the construction statement into a game winner for weak unforgeability that achieves the same performance.

Theorem 4.2. *Let* MAC *be a MAC scheme and* $\mathbf{mac} = (\mathsf{tag}, \mathsf{chk})$ *be the protocol based on* MAC *as described above. For the simulator* $\sigma_{\mathbf{mac}}$*,*

$$\left(\bullet\!\!= \!\!=\!\!\bullet^{\mathcal{K}}, \overset{\mathcal{M}\times\mathcal{T}}{-\twoheadrightarrow} \right) \quad \xrightarrow{\mathbf{mac}, \sigma_{\mathbf{mac}}, (0,\varepsilon)} \quad \bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow^{\mathcal{M}}.$$

The statement is parametrized in $q \in \mathbb{N}$ *for the resources* $\overset{q\ \mathrm{msgs}}{-\twoheadrightarrow}$ *and* $\bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow^{q\ \mathrm{msgs}}$*, and the performance measure* $\varepsilon_q := [\![\mathbf{G}_q^{\text{WUF-CMA}}(\mathrm{MAC})]\!] \circ \mathbf{C}_{\mathsf{mac}}$ *for a reduction* $\mathbf{C}_{\mathsf{mac}}$ *explicitly described in the proof.*

Proof. For the availability condition, we show

$$\mathsf{tag}^A \mathsf{chk}^B \left(\bullet\!\!= \!\!=\!\!\bullet_{\perp}, \overset{q\ \mathrm{msgs}}{-\twoheadrightarrow}_{\perp} \right) \quad \equiv \quad \bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow^{q\ \mathrm{msgs}}_{\perp}.$$

This follows directly from the correctness of the MAC scheme: for the key $k \in \mathcal{K}$ output by $\bullet\!\!= \!\!=\!\!\bullet$ and all messages $m \in \mathcal{M}$, $\mathrm{check}(k, m, \mathrm{mac}(k, m)) = \underline{\mathsf{true}}$ and hence chk accepts all tags generated by tag. In both cases, the system provides output until q messages have been input.

For the security condition, we define

$$\mathbf{R}_q \quad := \quad \mathsf{tag}^A \mathsf{chk}^B \left(\bullet\!\!= \!\!=\!\!\bullet, \overset{q\ \mathrm{msgs}}{-\twoheadrightarrow} \right) \qquad \text{and} \qquad \mathbf{S}_q \quad := \quad \sigma_{\mathbf{mac}}{}^E \bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow^{q\ \mathrm{msgs}},$$

which are random systems. We describe a reduction system $\mathbf{C}_{\mathsf{mac}}$ which at the left interface provides three sub-interfaces A, B, and E, and behaves as follows:

- On input a message $m \in \mathcal{M}$ at the left A-sub-interface, query $\mathbf{tag}(m)$ at the right interface to obtain a tag $t \in \mathcal{T}$. Output (m, t) at the left E-sub-interface.

- On input a pair $(\tilde{m}, \tilde{t})$ at the left E-sub-interface, query $\mathbf{vrf}\ (\tilde{m}, \tilde{t})$ at the right interface. If the return value is $\underline{\mathsf{true}}$, then output m at the B-sub-interface.

Analogously to the behavior of the protocol and the constructed channel, the converter $\mathbf{C}_{\mathsf{mac}}$ processes the inputs at the A- and E-sub-interfaces in order: the i-th input is processed only after all preceding inputs, i.e., those at sub-interfaces corresponding to a number j with $j < i$, have been processed. The converter $\mathbf{C}_{\mathsf{mac}}$ is equipped with the MBO that formalizes the restriction on the number of messages; in particular, the system also takes only q messages at the left A- and E-sub-interfaces before the MBO becomes 1. This reduction system satisfies $\mathbf{R}_q \equiv \mathbf{C}_{\mathsf{mac}}\mathbf{G}_q^{\text{WUF-CMA}}$, because both systems compute the tags and the MAC verification in the same way and output the same messages. Both systems provide output as long as at most q messages have been input at both the A- and E-interfaces.

Then we define an MBO $A_1, A_2, \ldots$ that becomes 1 as soon as an input $(\tilde{m}, \tilde{t}) \in \mathcal{M} \times \mathcal{T}$ is provided at the E-sub-interface such that $\tilde{m}$ was not input at the A-sub-interface before and $\mathrm{check}(k, \tilde{m}, \tilde{t}) = \underline{\mathsf{true}}$. This MBO can be applied to both systems $\mathbf{R}_q$ and $\mathbf{S}_q$, and the systems behave equivalently as long as the MBO remains 0. This holds because the outputs are computed in exactly the same way, and the systems provide output as long as at most q messages have been input at both the A- and E-interfaces. As the MBO $A_1, A_2, \ldots$ becomes 1 only exactly if the game $\mathbf{G}_q^{\text{WUF-CMA}}$ is won, we obtain

$$\hat{\mathbf{R}}_q \quad \stackrel{\mathrm{g}}{\equiv} \quad \hat{\mathbf{S}}_q \quad \stackrel{\mathrm{g}}{\equiv} \quad \mathbf{C}_{\mathsf{mac}}\mathbf{G}_q^{\text{WUF-CMA}},$$

and Lemmas 2.14 and 2.4 imply that

$$[\![(\mathbf{R}_q \mid \mathbf{S}_q)]\!] \quad \leq \quad [\![\mathbf{C}_{\mathsf{mac}}\mathbf{G}_q^{\text{WUF-CMA}}]\!] \quad = \quad [\![\mathbf{G}_q^{\text{WUF-CMA}}]\!] \circ (\cdot\mathbf{C}_{\mathsf{mac}}),$$

which concludes the proof. □

Weak unforgeability is considered the standard property for MAC schemes in the literature; our result shows that if the MAC scheme is used in the straightforward way, one indeed achieves a authentication guarantee in terms of the constructed channel.

4.3.3 Constructing Ordered Authenticated Channels

The channel $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$ constructed in the previous sub-sections is only of limited use for applications. The reason is that the only guarantee it formalizes is that any message that is delivered at the B interface has indeed been input

at the A-interface before, but there is no guarantee on the order in which the messages are delivered, or even on the number of times a certain message is delivered. This can be addressed by adding sequence numbers to the messages.

Consider the following protocol **seq** $= (\mathsf{seq}, \mathsf{seq\text{-}chk})$ that behaves as follows. Analogously to the converters described before, the converters process the messages in the order of their sub-interface identifiers, i.e., seq processes the inputs in the order $\mathsf{Out}.1?, \mathsf{Out}.2?, \ldots$, and seq-chk processes the received messages in the order $\mathsf{In}.1!, \mathsf{In}.2!, \ldots$.

- The converter seq, upon input the i-th message $m_i \leftarrow \mathsf{Out}.i?$ at the outside interface, outputs the pair (i, m_i) at the inside interface $\mathsf{In}.i?$.
- The converter seq-chk keeps a counter $i \in \mathbb{N}$ which is initialized to 0. Also, it keeps a buffer $\mathcal{B}$ which is initialized to be empty and stores pairs in $\mathbb{N} \times \mathcal{M}$. Upon input a pair (j, m) at the inside interface: If $j > i + 1$ and there is no element $(j, m') \leftarrow \mathsf{In}.k! \in \mathcal{B}$, once $\mathsf{In}.l! \neq \square$ for all $l \leq k$, set $\mathcal{B} \leftarrow \mathcal{B} \cup \{(j, m)\}$. If $j = i + 1$, then set $i \leftarrow j$ and output m at $\mathsf{Out}.j!$. Also, while there is an element $(i + 1, m') \in \mathcal{B}$, output m', set $\mathcal{B} \leftarrow \mathcal{B} \setminus \{(i + 1, m')\}$ and set $i \leftarrow i + 1$.

Out goal is to show that the protocol **seq** constructs the channel $\bullet\!\!-\!\twoheadrightarrow$ from the channel $\bullet\!\!-\!\diamond\!\!-\!\twoheadrightarrow$. This holds with respect to the simulator σ_{seq}, which behaves as follows. σ_{seq} keeps a counter $n \in \mathbb{N}$ (initially 0) and a buffer $\mathcal{B}$ (initially empty) for keeping track of messages that are still to be delivered. On input the i-th message $m_i \leftarrow \mathsf{In}.i!$ at the inside interface, σ_{seq} outputs (i, m_i) at the outside interface $\mathsf{Out}.i!$. Upon an input $k \in \mathbb{N}$ at the outside interface $\mathsf{Out}.i?$, if $k > n + 1$ then set $\mathcal{B} \leftarrow \mathcal{B} \cup \{k\}$. If $k = n + 1$, output $\uparrow$ at the k-th inside sub-interface $\mathsf{In}.k?$ and set $n \leftarrow k$. Also, while there is an element $(n + 1, m') \in \mathcal{B}$, output $\uparrow$ at the $(n + 1)$-th inside sub-interface $\mathsf{In}.(n + 1)?$, set $n \leftarrow n + 1$.

Theorem 4.3. *Let* **seq** $= (\mathsf{seq}, \mathsf{seq\text{-}chk})$ *be the protocol described above. For the simulator* σ_{seq}*,*

$$\bullet\!\!-\!\diamond\!\!-\!\overset{\mathbb{N}\times\mathcal{M}}{\twoheadrightarrow} \quad \xRightarrow{\mathbf{seq},\sigma_{\mathsf{seq}},(0,0)} \quad \bullet\!\!-\!\overset{\mathcal{M}}{\twoheadrightarrow},$$

where the statement is parametrized in $q \in \mathbb{N}$ *for the resources* $\bullet\!\!-\!\diamond\!\!-\!\overset{q\text{ msgs}}{\twoheadrightarrow}$ *and* $\bullet\!\!-\!\overset{q\text{ msgs}}{\twoheadrightarrow}$.

Proof. For the availability condition, we show

$$\mathsf{seq}^A \mathsf{seq\text{-}chk}^B \bullet\!\!-\!\diamond\!\!-\!\overset{q\text{ msgs}}{\twoheadrightarrow}_{\perp} \quad \equiv \quad \bullet\!\!-\!\overset{q\text{ msgs}}{\twoheadrightarrow}_{\perp}.$$

This is immediate because the channel $\bullet\!\!-\!\diamond\!\!-\!\twoheadrightarrow_{\perp}$ will (as $\bullet\!\!-\!\twoheadrightarrow$) deliver all messages, and the sequence numbers generated by seq will be checked successfully (and removed) by seq-chk.

For the security condition, we define

$$\mathbf{R}_q \quad \coloneqq \quad \mathsf{seq}^A \mathsf{seq\text{-}chk}^B \bullet\!\!-\!\!\overset{q\ \text{msgs}}{\diamond}\!\!\twoheadrightarrow \qquad \text{and} \qquad \mathbf{S}_q \quad \coloneqq \quad \sigma_{\mathsf{seq}}{}^E \bullet\!\!-\overset{q\ \text{msgs}}{}\!\!\twoheadrightarrow.$$

The two systems can be seen to be equivalent as follows: The i-th input message m_i at the A-interface will lead to an output (i, m_i) at the E-interface in both cases. Also, an input $j \in \mathbb{N}$ at the E-interface will lead to the same results:

- If the j-th message has already been delivered to B, then no message will be output.
- If the $(j-1)$-th message has not yet been delivered to B, then there is no immediate output.
- Otherwise, the j-th message m_j is output at the B-interface. Moreover, if inputs $j+1, j+2, \ldots$ have already been provided before, the corresponding messages $m_{j+1}, m_{j+2}, \ldots$ are also output at the B-interface.

This concludes the proof. □

4.4 Symmetric Encryption

Symmetric encryption protects the confidentiality of messages transmitted between two parties that share a secret key. Intuitively, this means that the encrypted message (the ciphertext) transmitted from a sender A to a receiver B does not leak information about the contents of the message (other than, for example, its length). The natural application of a symmetric encryption scheme is to construct a secure channel from an authenticated channel and a shared secret key.

4.4.1 The One-Time Pad and Stream Ciphers

The one-time pad encryption scheme is defined on bit strings. A random key, which is a bit string of the same length as the message, is added to the message bit-by-bit. The first proof of the one-time pad has been given by Shannon [Sha49], who showed that the ciphertext, as a random variable, is independent of the plaintext and therefore does not contain any information about the message. The constructive perspective on the one-time pad has been described by Maurer [Mau11, Mau14]; we formalize the corresponding proof in the formal framework introduced here.

We describe the one-time pad for a single message of arbitrary length. The protocol can be viewed as assuming two resources: First, a stream of key bits which are described as $\langle \bullet\!\overset{\text{1-bit}}{=\!=}\!\bullet \rangle$, that is, an unbounded sequence of uniformly random bits shared by A and B. The second assumed resource is a channel over which the ciphertext is transmitted from the sender A to the receiver B. The message space of this channel is $\{0,1\}^*$. The protocol based on the one-time pad is then described as a pair $\mathbf{otp} = (\mathsf{otp\text{-}enc}, \mathsf{otp\text{-}dec})$ of converters for the sender and the receiver. These converters behave as follows.

$\mathsf{otp\text{-}enc}$**:** The converter takes at the outside interface $\mathsf{Out}.1?$ an input $m \in \{0,1\}^*$ and retrieves at the first inside sub-interface $|m|$ key bits (by providing input at $\mathsf{In}.1?, \ldots, \mathsf{In}.|m|?$), call the concatenation of the obtained bits k. It then computes $c \leftarrow m \oplus k$ and outputs it at the inside sub-interface $\mathsf{In}.2.1?$.

$\mathsf{otp\text{-}dec}$**:** The converter $\mathsf{otp\text{-}dec}$ receives a ciphertext c' at its inside sub-interface $\mathsf{In}.2.1!$ and retrieves at the first inside sub-interface $|c'|$ key bits analogously to $\mathsf{otp\text{-}enc}$, call the concatenation k'. It then computes $m' \leftarrow c' \oplus k'$ and outputs it at the outside interface.

The authenticated case. If the ciphertext is transmitted via an authenticated communication channel, then the one-time pad indeed achieves secure communication [Mau11]. Constructively, this is phrased as

$$\left(\left\langle \bullet\!\overset{\text{1-bit}}{=\!=}\!\bullet \right\rangle, \bullet\!\!- \rightarrow\right) \quad \xrightarrow{\mathbf{otp}} \quad \bullet\!\!- \rightarrow\!\!\bullet.$$

The constructed secure channel leaks the length of the transmitted message, which is due to the fact that the message and the ciphertext have the same length, and shows up in the following security proof because the simulator has to generate a ciphertext that has the same distribution as the ciphertext generated by the encryption converter. The application of the one-time pad in the described scenario is depicted in detail in Figure 4.4.

The construction is shown relative to the simulator $\sigma_{\mathbf{otp}}$, which on input $n \in \mathbb{N}$ at the inside interface (and once sufficiently many trigger inputs $\mathsf{Out}.1.1.A?, \ldots, \mathsf{Out}.1.|m|.A?$ have been provided for the key bits), outputs a random string $\tilde{c} \in \{0,1\}^n$ at the outside interface. The trigger input $\uparrow$ at the second outside sub-interface $\mathsf{Out}.2.1?$ (i.e., corresponding to $\bullet\!\!- \rightarrow$) is forwarded to the inside interface $\mathsf{In}.2.1?$ (i.e., to $\bullet\!\!- \rightarrow\!\!\bullet$), once the trigger inputs $\mathsf{Out}.1.1.B?, \ldots, \mathsf{Out}.1.|m|.B?$ have been provided for the key bits. The setting with the constructed channel $\bullet\!\!- \rightarrow\!\!\bullet$ and the described simulator $\sigma_{\mathbf{otp}}$ is depicted in Figure 4.5.

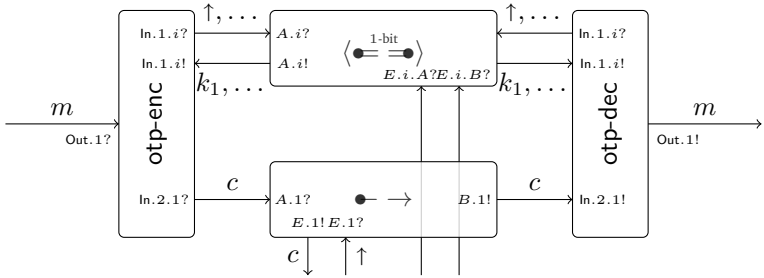


Figure 4.4: The message flow in the one-time-pad-based protocol.

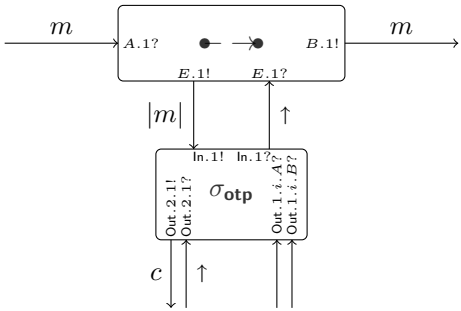


Figure 4.5: The message flow in the execution with the simulator.

The following theorem shows that the one-time-pad-based protocol indeed achieves the described construction.

Theorem 4.4. *Let* **otp** $=$ (otp-enc, otp-dec) *be the protocol described above. For the simulator* $\sigma_{\mathbf{otp}}$*,*

$$\left(\left\langle \bullet\!\overset{\text{1-bit}}{=\!=}\!\bullet \right\rangle, \bullet\!\leftarrow\!\rightarrow\right) \xrightarrow{\mathbf{otp},\sigma_{\mathbf{otp}},(0,0)} \bullet\!\leftarrow\!\rightarrow\!\bullet,$$

which is a statement parametrized in $\ell \in \mathbb{N}$ *for the resources* $\langle \bullet\!\overset{\text{1-bit}}{=\!=}\!\bullet \rangle_{[\ell]}$, $\bullet\!\overset{\ell\text{ bits}}{\leftarrow}\!\rightarrow$, *and* $\bullet\!\overset{\ell\text{ bits}}{\leftarrow}\!\rightarrow\!\bullet$.

Proof. For the availability condition, we show that

$$\text{otp-enc}^A\text{otp-dec}^B\left(\left\langle \bullet\!\overset{\text{1-bit}}{=\!=}\!\bullet_{\perp} \right\rangle_{[\ell]}, \bullet\!\overset{\ell\text{ bits}}{\leftarrow}\!\rightarrow_{\perp}\right) \equiv \bullet\!\overset{\ell\text{ bits}}{\leftarrow}\!\rightarrow\!\bullet_{\perp}.$$

This is easy to see since for a message m and the first $|m|$ key bits $k = k_1 \cdots k_{|m|}$, otp-enc computes $c = m \oplus k$, and otp-dec computes $m' = c \oplus k =$

$m \oplus k \oplus k = m$. Hence, on input a message m at the A-interface, both systems output m at the B-interface. In both cases, the system provides output unless the message input is longer than ℓ bits.

For the security condition, we show that

$$\mathsf{otp\text{-}enc}^A \mathsf{otp\text{-}dec}^B \left(\left\langle \overset{\text{1-bit}}{\bullet\!\!=\!\!=\!\!\bullet} \right\rangle_{[\ell]}, \overset{\ell \text{ bits}}{\bullet\!\!-\!\!\rightarrow} \right) \quad \equiv \quad \sigma_{\mathbf{otp}}{}^E \overset{\ell \text{ bits}}{\bullet\!\!-\!\!\rightarrow\!\!\bullet}.$$

On input a message m at the A-interface (and once sufficiently many trigger inputs have peen provided for the key bits), both systems output a uniformly random $|m|$-bit string at the E-interface. This holds since otp-enc computes $c = m \oplus k$ with uniformly random bits $k = k_1 \cdots k_{|m|}$, and $\sigma_{\mathbf{otp}}$ chooses a fresh $|m|$-bit string. Upon input the trigger input "$\uparrow$" at the interface $E.2.1?$ (and once sufficiently many trigger inputs have peen provided for the key bits), both systems output m at the B-interface (by the same argument as for the availability). Also here, both systems provide output unless the message input is longer than ℓ bits. $\square$

As explained by Maurer [Mau11], the constructive proof of the one-time pad extends to practical encryption schemes such as stream ciphers and the counter-mode encryption of a block cipher since one can show that the key stream $\langle \overset{\text{1-bit}}{\bullet\!\!=\!\!=\!\!\bullet} \rangle$ is constructed by such schemes. The composition theorem then allows to conclude the security of these "computational" schemes.

The unauthenticated case. In principle, one could send the ciphertext generated by the one-time pad over an insecure channel. If the ciphertext, however, is modified during the transmission, the receiver will not detect this modification (since every bit string is a valid ciphertext). The decryption will then result a message that is different from the one input by the sender. Concretely, if an attacker replaces a ciphertext c by a different ciphertext c', then the receiver will compute $m' = k \oplus c'$, and as by the definition of the encryption $k = m \oplus c$, the computed message is $m' = (m \oplus c) \oplus c' = m \oplus (c \oplus c')$. This means that the attacker can, by replacing the ciphertext, add an XOR mask $c \oplus c'$ of his choice to the transmitted message. We formalize the obtained guarantee by the *XOR-malleable confidential channel*, denoted $\rightarrow\!\!\oplus\!\!\rightarrow\!\!\bullet$, which is confidential but allows to input a mask at the E-interface, which is then added to the transmitted message. The one-time pad hence also achieves the construction

$$\left(\left\langle \overset{\text{1-bit}}{\bullet\!\!=\!\!=\!\!\bullet} \right\rangle, - \rightarrow \right) \quad \xrightarrow{\mathbf{otp}} \quad \rightarrow\!\!\oplus\!\!\rightarrow\!\!\bullet,$$

which we describe here in more detail.

The XOR-malleable confidential channel $\xrightarrow{\oplus}\bullet$ has been described for fixed-length messages in Section 4.1.4, for the case of variable-length messages the channel is specified as System 25.[4] A message $m \in \{0,1\}^n$ from the sender is encrypted by adding the key $k \in \{0,1\}^n$ bit-by-bit, resulting in the ciphertext $c = m \oplus k \in \{0,1\}^n$. If the attacker replaces the ciphertext c by $c' \in \{0,1\}^{n'}$, where possibly $n \neq n'$, then the first $\min(n, n')$ bits obtained by the receiver are the XOR of the corresponding bits of m, c, and c'. If $n' > n$, then the subsequent bits are uniformly random.

In an execution of the protocol based on the one-time pad, the attacker can also inject a message to the receiver B before the sender A has input a message. In that case, for an injected ciphertext c' of length $n' = |c'|$, the receiver B will obtain an n'-bit uniformly random string m'. If later the sender inputs a message m of length n, then the output at the E-interface is computed to be consistent with the output at the B-interface. (Recall that, intuitively, the inputs and outputs have to satisfy that $m \oplus c = k = m' \oplus c'$.)

The described construction is achieved relative to the simulator σ'_{otp} that simply forwards the messages between the inside and the outside interface. The construction is again achieved in a perfect sense.

Theorem 4.5. *Let* $\mathbf{otp} = (\mathsf{otp\text{-}enc}, \mathsf{otp\text{-}dec})$ *be the protocol described above. For the simulator* σ'_{otp}*,*

$$\left(\left\langle \overset{\text{1-bit}}{\bullet\!\!=\!\!=\!\!\bullet} \right\rangle, \rightarrow\right) \xrightarrow{\mathbf{otp},\sigma'_{\mathsf{otp}},(0,0)} \xrightarrow{\oplus}\bullet,$$

which is a statement parametrized in $\ell \in \mathbb{N}$ *for the resources* $\langle \overset{\text{1-bit}}{\bullet\!\!=\!\!=\!\!\bullet} \rangle_{[\ell]}$, $\overset{\ell \text{ bits}}{\rightarrow}$, *and* $\overset{\ell \text{ bits}}{\xrightarrow{\oplus}}\bullet$.

Proof. For the availability condition, we show that

$$\mathsf{otp\text{-}enc}^A \mathsf{otp\text{-}dec}^B \left(\left\langle \overset{\text{1-bit}}{\bullet\!\!=\!\!=\!\!\bullet}\bot \right\rangle_{[\ell]}, \overset{\ell \text{ bits}}{\rightarrow}\bot\right) \quad \equiv \quad \overset{\ell \text{ bits}}{\xrightarrow{\oplus}}\bullet\bot.$$

This follows exactly as in Theorem 4.4. For the security condition, we show that

$$\mathsf{otp\text{-}enc}^A \mathsf{otp\text{-}dec}^B \left(\left\langle \overset{\text{1-bit}}{\bullet\!\!=\!\!=\!\!\bullet} \right\rangle_{[\ell]}, \overset{\ell \text{ bits}}{\rightarrow}\right) \quad \equiv \quad \sigma'_{\mathsf{otp}}{}^E \overset{\ell \text{ bits}}{\xrightarrow{\oplus}}\bullet.$$

We differentiate two cases:

[4] The fact that the resource is indeed a monotone discrete system can be verified by providing an alternative, equivalent description in which a bit string resembling the key is sampled initially, and the outputs are computed as in the protocol.

System 25 XOR-malleable single-use channel $\multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$

```
upon A.1?
    if E.1? ≠ □ then
        if |A.1?| > |E.1?| then
            x ←$ {0,1}^(|A.1?|−|E.1?|)
        end if
        x ← (E.1? ⊕ B.1!).x
        E.1! ← A.1? ⊕ (x_1 ··· x_|A.1?|)
    else
        E.1! ←$ {0,1}^|A.1?|
    end if
end.
upon E.1?
    if A.1? ≠ □ then
        if |E.1?| > |A.1?| then
            x ←$ {0,1}^(|E.1?|−|A.1?|)
        end if
        x ← (A.1? ⊕ E.1!).x
        B.1! ← E.1? ⊕ (x_1 ··· x_|E.1?|)
    else
        B.1! ←$ {0,1}^|E.1?|
    end if
end.
```

- If the first input is a message $m \in \{0,1\}^*$ at the A-interface, both systems output a uniformly random $|m|$-bit string at the E-interface. This holds since otp-enc computes $c = m \oplus k$ with uniformly random bits $k = k_1 \cdots k_{|m|}$, and $\multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$ chooses a fresh $|m|$-bit string. Upon input a string $\tilde{c}$ at the E-interface:
  - If $|m| \geq |\tilde{c}|$, then both systems output $(m_1 \cdots m_{|\tilde{c}|}) \oplus (c_1 \cdots c_{|\tilde{c}|}) \oplus \tilde{c}$ at the B-interface. This follows since $(m_1 \cdots m_{|\tilde{c}|}) \oplus (m_1 \cdots m_{|\tilde{c}|}) = k_1 \cdots k_{|\tilde{c}|}$, which is what otp-dec computes, and the same computation is performed by $\multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$.
  - Otherwise, the first $|m|$ bits are computed as above, and are output at the B-interface together with $|\tilde{c}| - |m|$ random bits. In otp-dec, this holds because $\tilde{c}_{|c|+1} \cdots \tilde{c}_{|\tilde{c}|}$ are XORed with uniformly random key bits. In ${\sigma'_{\mathsf{otp}}}^{E} \multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$, these bits are chosen uniformly at random by $\multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$.

- If the first input is a ciphertext $\tilde{c}$ at the E-interface, both systems output a uniformly random $|\tilde{c}|$-bit string at the B-interface. This holds since otp-dec computes $m' = \tilde{c} \oplus k$ with uniformly random bits $k = k_1 \cdots k_{|\tilde{c}|}$, and $\multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$ chooses a fresh $|\tilde{c}|$-bit string (upon input from σ'_{otp}). Upon input a string m at the E-interface:
  - If $|\tilde{c}| \geq |m|$, then both systems output $(\tilde{c}_1 \cdots \tilde{c}_{|m|}) \oplus (m'_1 \cdots m'_{|m|}) \oplus m$ at the E-interface. This follows since

    $$(\tilde{c}_1 \cdots \tilde{c}_{|m|}) \oplus (m'_1 \cdots m'_{|m|}) \;=\; (k_1 \cdots k_{|m|}),$$

    which is what otp-enc computes, and the same computation is performed by $\multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$.
  - Otherwise, the first $|\tilde{c}|$ bits are computed as above, and are output at the E-interface together with $|m| - |\tilde{c}|$ random bits. In otp-enc, this holds because the remaining bits of the message are XORed with uniformly random bits. In ${\sigma'_{\mathsf{otp}}}^{E} \multimap\!\!\oplus\!\!\rightarrow\!\!\bullet$, the remaining bits are chosen uniformly at random.

Also here, both systems provide output unless the message input is longer than ℓ bits. □

The XOR-malleable channel is of course not immediately useful for applications. We show in Section 4.5.2, however, that a strongly unforgeable MAC can be used to construct a secure channel from the XOR-malleable channel and a shared secret key.

4.4.2 CBC-Mode Encryption

The cipher-block chaining (CBC) encryption mode of a block cipher is a widely used encryption scheme. For instance, the widely used protocols SSL/TLS, ssh, and IPsec all specify encryption modes based on CBC, despite the fact that other modes such as the counter mode provide (at least) the same level of security and, depending on the exact application, sometimes even require weaker assumptions on the underlying primitives (no inversion queries to the block cipher). CBC-mode encryption has first been formally analyzed by Bellare et al. [BDJR97] in a game-based security model, and a different way to obtain the construction statement we show in this section is to use the result of [BDJR97] together with Theorem 4.7 from Section 4.4.3.

The CBC-mode encryption scheme is described as a pair of converters $\mathbf{cbc}_n = (\mathsf{cbc\text{-}enc}_n, \mathsf{cbc\text{-}dec}_n)$ that assumes as resources a shared n-bit uniform random permutation $\overleftrightarrow{\mathbf{P}}_n$ (i.e., the permutation is over $\{0,1\}^n$) and a channel with message space $\{0,1\}^\ell$ for $\ell \in n \cdot \mathbb{N}$. The channel constructed by CBC-mode encryption also transmits messages in $\{0,1\}^\ell$ for $\ell \in n \cdot \mathbb{N}$; each ciphertext is n bits longer than the corresponding plaintext. We show that CBC achieves the construction

$$\left(\overleftrightarrow{\mathbf{P}}, \bullet\!\!- \rightarrow\right) \xrightarrow{\mathbf{cbc}_n} \bullet\!\!- \rightarrow\!\!\bullet.$$

An n-bit URP (as a random system) allows the permutation to be evaluated both in the forward and in the backward direction. This is achieved by describing it as a random system $\mathbf{P}_n$ as follows: Based on a uniform random permutation $P_n : \{0,1\}^n \rightarrow \{0,1\}^n$, if an input $X_i = (\mathsf{fwd}, x)$ for some $x \in \{0,1\}^n$ is given, then $\mathbf{P}_n$ responds with $Y_i = P_n(x)$. If an input $X_i = (\mathsf{bwd}, y)$ for some $y \in \{0,1\}^n$ is given, then $\mathbf{P}_n$ responds with $Y_i = P_n^{-1}(y)$. The shared URP $\overleftrightarrow{\mathbf{P}}_n$ is then obtained as described in Section 4.2.

The converters $\mathsf{cbc\text{-}enc}_n$ and $\mathsf{cbc\text{-}dec}_n$ behave as follows. Denote by $m = m_1.m_2.\ldots$ the plaintext message consisting of n-bit blocks $m_i \in \{0,1\}^n$ and input at interface A, and by $c = c_0.c_1.\ldots$ the generated ciphertext. The ciphertext received by B are denoted by $c' = c'_0.c'_1.\ldots$, and the decrypted plaintext by $m' = m'_1.m'_2.\ldots$.

$\mathsf{cbc\text{-}enc}_n$**:** Upon input a message m at the outside interface, the encryption is computed iteratively by choosing an IV $c_0 \in_R \{0,1\}^n$ uniformly at random and then iteratively querying, for each $i \in [|m|/n]$, at the inside interface $\mathsf{In}.1.i? \leftarrow (\mathsf{fwd}, c_{i-1} \oplus m_i)$, obtaining as response $c_i \leftarrow \mathsf{In}.1.i!$. The ciphertext $c = c_0.c_1.\ldots$ is output at the inside sub-interface $\mathsf{In}.1.1?$.

$\mathsf{cbc\text{-}dec}_n$**:** Upon input a ciphertext c' at the second inside sub-interface, the decryption is performed by iteratively querying, for each $i \in [|c'|/n -$

1], at the inside interface $\mathsf{In}.1.i? \leftarrow (\mathsf{bwd}, c'_i)$, and after obtaining the response, computing $m'_i \leftarrow c'_{i-1} \oplus \mathsf{In}.1.i!$. The plaintext $m' = m'_1.m'_2.\ldots$ is output at the outside interface $\mathsf{Out}.1!$.

If CBC-mode encryption is applied to an authenticated channel, then the protocol indeed constructs a secure channel. In contrast to the one-time pad, the construction is not perfect, but one obtains an error term that originates from possible collisions in the computation.

We prove the construction relative to a simulator $\sigma_{\mathbf{cbc}}$ that, on input a message length $l \in n \cdot \mathbb{N}$ at the inside interface, outputs a uniformly random $(l+n)$-bit string at the outside sub-interface $\mathsf{Out}.2.1!$. On input the trigger input $\uparrow$ at the outside sub-interface $\mathsf{Out}.2.1?$, $\sigma_{\mathbf{cbc}}$ outputs $\uparrow$ at the inside interface $\mathsf{In}.1?$.

Theorem 4.6. *For the protocol* $\mathbf{cbc}_n = (\mathsf{cbc\text{-}enc}_n, \mathsf{cbc\text{-}dec}_n)$ *and the simulator* $\sigma_{\mathbf{cbc}}$*,*

$$\left(\overleftrightarrow{\mathbf{P}}_n, \bullet\!\!\leftarrow\!\!\rightarrow\right) \quad \xRightarrow{\mathbf{cbc}_n, \sigma_{\mathbf{cbc}}, (\ell/n)^2/2^n} \quad \bullet\!\!\leftarrow\!\!\rightarrow\!\!\bullet,$$

which is a statement parametrized in $\ell \in n \cdot \mathbb{N}$ *for the resources* $\overleftrightarrow{\mathbf{P}}_n^{|\ell/n}$, $\bullet\!\!\overset{\ell+n \text{ bits}}{\leftarrow}\!\!\rightarrow$, *and* $\bullet\!\!\overset{\ell \text{ bits}}{\leftarrow}\!\!\rightarrow\!\!\bullet$.

Proof. For the availability condition, we show that

$$\mathsf{cbc\text{-}enc}_n{}^A \mathsf{cbc\text{-}dec}_n{}^B \left(\overleftrightarrow{\mathbf{P}}_n^{|\ell/n}{}_{\perp}, \bullet\!\!\overset{(\ell+1)n \text{ bits}}{\leftarrow}\!\!\rightarrow_{\perp}\right) \quad \equiv \quad \bullet\!\!\overset{\ell n \text{ bits}}{\leftarrow}\!\!\rightarrow\!\!\bullet_{\perp}.$$

Denote the IV as c_0, and the instance of the uniform random permutation by $P_n : \{0,1\}^n \to \{0,1\}^n$. The ciphertext for the message m is then computed as $c_0.\ldots.c_l$ with $l = \frac{|m|}{n}$, with $c_j = P_n(c_{j-1} \oplus m_j)$. The decryption then computes, for each $j \in [l]$, $m'_j = c_{j-1} \oplus P_n^{-1}(c_j) = c_{j-1} \oplus c_{j-1} \oplus m_j = m_j$, which means that on input m at the sender, the receiver will output m. This proves the correctness condition, since the constructed resource simply forwards the message and the MBO formalizing the usage restriction becomes 1 in the same cases.

For the security condition, we use the notation

$$\mathbf{R}_\ell \coloneqq \mathsf{cbc\text{-}enc}_n{}^A \mathsf{cbc\text{-}dec}_n{}^B \left(\overleftrightarrow{\mathbf{P}}_n, \bullet\!\!\leftarrow\!\!\rightarrow\right) \qquad \text{and} \qquad \mathbf{S}_\ell \coloneqq \sigma_{\mathbf{cbc}}{}^E \bullet\!\!\leftarrow\!\!\rightarrow\!\!\bullet$$

and consider a "hybrid" system $\mathbf{H}$ between $\mathbf{R}_\ell$ and $\mathbf{S}_\ell$ that generates the ciphertext at the E-interface similarly to $\mathbf{R}_q$, but instead of using a shared URP $\overleftrightarrow{\mathbf{P}}_n$, a shared URF $\overleftrightarrow{\mathbf{F}}_{n,n}$ is used.

We then define an MBO on $\mathbf{H}$ that becomes 1 once two output blocks at the E-interface collide non-trivially (that is, for m_i, m_j and c_{i-1}, c_{j-1} with $m_i \oplus c_{i-1} \neq m_j \oplus c_{j-1}$ it holds that $c_i = c_j$). As the output distributions of $\mathbf{R}_\ell$ and $\mathbf{H}$ are the same (they consist of strings which are uniform from the set of all strings for which no collision occurs), we obtain that $\hat{\mathbf{H}} \models \mathbf{R}_\ell$ and by Theorem 2.16 that

$$\left[\!\left[\left(\mathsf{cbc\text{-}enc}_n{}^A \mathsf{cbc\text{-}dec}_n{}^B \left(\bullet\!\!- \rightarrow, \overleftrightarrow{\mathbf{P}}_n\right) \;\middle|\; \mathbf{H}\right)\right]\!\right] \quad \leq \quad \frac{(\ell/n)^2}{2} \cdot 2^{-n}.$$

Then, we define an MBO corresponding to collisions on the input to the URF on $\mathbf{H}$ (that is, for m_i, m_j and c_{i-1}, c_{j-1} it holds that $m_i \oplus c_{i-1} = m_j \oplus c_{j-1}$), and obtain that $\hat{\mathbf{H}} \models \mathbf{S}_\ell$ (as the ciphertexts output in both cases are uniformly distributed bit strings of the same length), and use again Theorem 2.16 to obtain

$$\left[\!\left[\left(\mathbf{H} \mid \sigma_{\mathsf{cbc}}{}^E \bullet\!\!- \rightarrow\!\!\bullet\right)\right]\!\right] \quad \leq \quad \frac{(\ell/n)^2}{2} \cdot 2^{-n},$$

which together with the triangle inequality for the performance in the distinction problem concludes the proof. □

The shared URP $\overleftrightarrow{\mathbf{P}}_n$ can be constructed from a shared secret key using an n-bit block cipher, the security of this scheme then follows directly by the composition theorem. If the ciphertext is transmitted over an insecure communication channel, the constructed channel is still confidential but exhibits malleability which resembles the block structure of the encryption scheme. (This is due to the fact that the ciphertext is split into blocks in the decryption, and each block is processed separately.) Sending a CBC ciphertext insecurely, however, has shown to lead to vulnerabilities such as described by, for instance, Vaudenay [Vau02] and Albrecht et al. [APW09]. The malleability of this channel together with the particular application in the TLS record-layer protocol are described in Chapter 5.

4.4.3 Relation to Previous Security Notions

Various security definitions for symmetric encryption schemes exist in the literature. The usual approach is to define a property that a scheme may achieve, and that often corresponds to a kind of confidentiality or integrity guarantee. The property is then defined by means of a game (as discussed in Section 2.4.1) in which the adversary has access to oracles that allow to, for instance, encrypt plaintext messages, or to decrypt or check the validity of ciphertexts, sometimes with additional constraints on the number or order of queries. The adversary then has to achieve a certain winning condition which

may correspond to generating a ciphertext that satisfies a certain condition, or to distinguish two cases in which it is provided with different sets of oracles. For many of these notions, it is not clear which guarantees the proven schemes provide when the ciphertexts are transmitted over a certain type of network.

Recall that in Definition 2.18 we defined an encryption scheme as a pair $\mathrm{SC} = (\mathrm{enc}, \mathrm{dec})$ of (possibly probabilistic) functions $\mathrm{enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and $\mathrm{dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\diamond\}$. Such an encryption scheme can also be viewed as a pair $\mathbf{sc} = (\mathsf{enc}, \mathsf{dec})$ of converters as follows. As before, the inputs at the outside interface of enc and at the second inside sub-interface of dec are processed in order.

enc: The converter enc initially provides output at $\mathsf{In}.1.1?$ to retrieve the key $k \in \mathcal{K}$ (at interface $\mathsf{In}.1.1!$). Upon each input $m_i \leftarrow \mathsf{Out}.i? \in \mathcal{M}$ at the outside interface, enc computes $c_i \leftarrow\$ \mathrm{enc}(k, m_i)$ and outputs c_i at the inside sub-interface $\mathsf{In}.2.i?$.

dec: The converter dec initially provides output at $\mathsf{In}.1.1?$ to retrieve the key $k \in \mathcal{K}$ (at interface $\mathsf{In}.1.1!$). Upon each input $c_i \leftarrow \mathsf{In}.2.i! \in \mathcal{C}$ at the inside interface, dec computes $m_i \leftarrow \mathrm{dec}(k, c_i)$ and outputs m_i at the outside interface $\mathsf{Out}.i!$.

The converters are supposed to connect with the first inside sub-interface to a key (such as $\bullet\!= =\!\bullet$) with key space $\mathcal{K}$, and with the second inside sub-interface to a (sequence of) channel(s).

We relate several previous definitions to constructions. In particular, we show that an encryption scheme SC satisfies IND-CPA if and only if the derived protocol $\mathbf{sc}$ achieves the construction

$$(\bullet\!= =\!\bullet, \langle \bullet\!- \!\rightarrow \rangle) \quad \xrightarrow{\mathbf{sc}} \quad \bullet\!- \!\rightarrow\!\bullet.$$

If the scheme SC satisfies even IND-CCA, then the protocol $\mathbf{sc}$ achieves the construction

$$(\bullet\!= =\!\bullet, - \twoheadrightarrow) \quad \xrightarrow{\mathbf{sc}} \quad -\!\diamond\!\twoheadrightarrow\!\bullet,$$

that is, it constructs a non-malleable unordered confidential channel. In this case, the converse statement does not hold because of an artificial strictness of the IND-CCA property. Last, we show that if the encryption scheme SC satisfies both IND-CPA and INT-CTXT, then the protocol $\mathbf{sc}$ achieves the construction

$$(\bullet\!= =\!\bullet, - \twoheadrightarrow) \quad \xrightarrow{\mathbf{sc}} \quad \bullet\!-\!\diamond\!\twoheadrightarrow\!\bullet,$$

that is, the protocol even constructs an unordered secure channel. As in the case of IND-CCA, the converse does not hold because INT-CTXT is artificially strict in the same sense as IND-CCA. The material in this section is taken from Maurer et al. [MRT12].

Indistinguishability of Ciphertexts under Chosen-Plaintext Attack—IND-CPA

The most widely used security notion for confidentiality of a symmetric encryption scheme is IND-CPA, i.e., indistinguishability (of ciphertexts) under chosen-plaintext attack. Several variants appear in the literature, as described in Section 2.4.3. We first show that any encryption scheme that satisfies IND-CPA gives rise to a protocol that constructs a secure channel from a shared secret key and an authenticated channel. We model the fact that the scheme may be used to encrypt an arbitrary (unbounded) number of messages by describing the construction with respect to a sequence $\langle \bullet\!\!- \rightarrow \rangle$ of authenticated channels and a sequence $\langle \bullet\!\!- \rightarrow\!\!\bullet \rangle$ of secure channels.

The construction is shown with respect to a simulator σ_{CPA} that initially chooses a key $k \leftarrow\!\!\$\, \mathcal{K}$. Upon receiving an integer $n \in \mathbb{N}$ at the inside interface (which describes the length of a transmitted message), σ_{CPA} chooses $m \leftarrow\!\!\$\, \{0,1\}^n$, computes $c \leftarrow\!\!\$\, \mathrm{enc}\,(k, m)$ and outputs c at the second outside sub-interface. Upon a trigger input at the second outside sub-interface, σ_{CPA} outputs the trigger value at the inside interface.

Theorem 4.7. *Let* $\mathrm{SC} = (\mathrm{enc}, \mathrm{dec})$ *be an encryption scheme and* **sc** *be the pair of converters described above. For the simulator* σ_{CPA}*,*

$$\left(\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet, \langle \bullet\!\!- \rightarrow \rangle\right) \quad \xrightarrow{\mathbf{sc},\sigma_{\text{CPA}},(0,\varepsilon)} \quad \langle \bullet\!\!- \rightarrow\!\!\bullet \rangle\,,$$

where the symbols refer to sequences $\langle \bullet\!\!- \rightarrow \rangle_{[q]}$ *and* $\langle \bullet\!\!- \rightarrow\!\!\bullet \rangle_{[q]}$*, and performance measures* $\varepsilon_q = [\![\big(\mathbf{G}^{\text{IND-CPA}}_{0,q}(\mathrm{SC}) \mid \mathbf{G}^{\text{IND-CPA}}_{1,q}(\mathrm{SC}) \big)]\!] \circ (\cdot \mathbf{C}_{\text{CPA}})$ *for a reduction converter* $\mathbf{C}_{\text{CPA}}$ *described in the proof.*

Proof. The availability condition means that for each $q \in \mathbb{N}$, the equation

$$\mathsf{enc}^A \mathsf{dec}^B \left(\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet, \langle \bullet\!\!- \rightarrow_{\perp} \rangle_{[q]}\right) \quad \equiv \quad \langle \bullet\!\!- \rightarrow\!\!\bullet_{\perp} \rangle_{[q]}$$

holds. This follows directly from the correctness of the encryption scheme according to Definition 2.18, and by the fact that both systems stop after receiving exactly q inputs at the A-interface.

For the security condition, we describe a converter $\mathbf{C}_{\text{CPA}}$ such that

$$\left(\mathsf{enc}^A\mathsf{dec}^B\left(\bullet\!\overset{\mathcal{K}}{=}\!\bullet, \langle\bullet\!\!-\!\!\rightarrow\rangle_{[q]}\right) \;\middle|\; {\sigma_{\text{CPA}}}^E\,\langle\bullet\!\!-\!\!\twoheadrightarrow\!\bullet\rangle_{[q]}\right)$$
$$= \quad \left(\mathbf{C}_{\text{CPA}}\mathbf{G}^{\text{IND-CPA}}_{0,q}(\text{SC}) \;\middle|\; \mathbf{C}_{\text{CPA}}\mathbf{G}^{\text{IND-CPA}}_{1,q}(\text{SC})\right).$$

The converter $\mathbf{C}_{\text{CPA}}$ has three outside sub-interfaces (labeled A, B, and E) and behaves as follows: Upon input the i-th message m_i at the A-sub-interface, $\mathbf{C}_{\text{CPA}}$ queries $\mathbf{enc}(m_i)$ at the inside interface, obtaining a ciphertext c_i which it outputs at the outside E-sub-interface. Upon input the i-th trigger input at the outside E-sub-interface and once i messages have been input at the A-sub-interface, $\mathbf{C}_{\text{CPA}}$ outputs m_i at the outside B-sub-interface.

Since

$$\mathsf{enc}^A\mathsf{dec}^B\left(\bullet\!\overset{\mathcal{K}}{=}\!\bullet, \langle\bullet\!\!-\!\!\rightarrow\rangle_{[q]}\right) \quad\equiv\quad \mathbf{C}_{\text{CPA}}\mathbf{G}^{\text{IND-CPA}}_{0,q}(\text{SC})$$

and

$${\sigma_{\text{CPA}}}^E\,\langle\bullet\!\!-\!\!\twoheadrightarrow\!\bullet\rangle_{[q]} \quad\equiv\quad \mathbf{C}_{\text{CPA}}\mathbf{G}^{\text{IND-CPA}}_{1,q}(\text{SC}),$$

the statement follows by Lemma 2.6. □

As a direct corollary, we obtain that the same protocol constructs the multi-message secure channel $\bullet\!\!-\!\!\twoheadrightarrow\!\bullet$ from the corresponding authenticated channel $\bullet\!\!-\!\!\twoheadrightarrow$ and a secret key, the statement holds with respect to a simulator σ'_{CPA} that behaves almost as σ_{CPA} but can be restricted to the case where the messages are delivered in the correct order. The same statement holds with respect to the unordered versions $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$ and $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\bullet$, again with a slightly modified simulator σ''_{CPA}.

Corollary 4.8. *Let* $\text{SC} = (\text{enc}, \text{dec})$ *be an encryption scheme and* **sc** *be the pair of converters described above. For the simulator* σ'_{CPA}*,*

$$\left(\bullet\!\overset{\mathcal{K}}{=}\!\bullet, \bullet\!\!-\!\!\twoheadrightarrow\right) \quad \xRightarrow{\mathbf{sc},\sigma'_{\text{CPA}},(0,\varepsilon)} \quad \bullet\!\!-\!\!\twoheadrightarrow\!\bullet,$$

where the symbols refer to sequences $\overset{q\text{ msgs}}{\bullet\!\!-\!\!\twoheadrightarrow}$ *and* $\overset{q\text{ msgs}}{\bullet\!\!-\!\!\twoheadrightarrow\!\bullet}$*, and*

$$\varepsilon_q = \left(\mathbf{G}^{\text{IND-CPA}}_{0,q}(\text{SC}) \;\middle|\; \mathbf{G}^{\text{IND-CPA}}_{1,q}(\text{SC})\right) \circ (\cdot\mathbf{C}'_{\text{CPA}})$$

for a reduction $\mathbf{C}'_{\text{CPA}}$ *similar to the one in Theorem 4.7. Analogously, for the simulator* σ''_{CPA}*,*

$$\left(\bullet\!\overset{\mathcal{K}}{=}\!\bullet, \bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\right) \quad \xRightarrow{\mathbf{sc},\sigma''_{\text{CPA}},(0,\varepsilon)'} \quad \bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\bullet,$$

where the symbols refer to sequences $\overset{q\text{ msgs}}{\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow}$ *and* $\overset{q\text{ msgs}}{\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\bullet}$*, and*

$$\varepsilon'_q = \left(\mathbf{G}^{\text{IND-CPA}}_{0,q}(\text{SC}) \;\middle|\; \mathbf{G}^{\text{IND-CPA}}_{1,q}(\text{SC})\right) \circ (\cdot\mathbf{C}''_{\text{CPA}})$$

for a reduction $\mathbf{C}''_{\text{CPA}}$ *similar to the one in Theorem 4.7.*

The condition of achieving IND-CPA is not only sufficient but also necessary for the construction to be valid. If the protocol $\mathbf{sc} = (\mathsf{enc}, \mathsf{dec})$ achieves the construction shown in Theorem 4.7 with respect to the simulator σ_{CPA} described there, then the underlying symmetric encryption scheme SC satisfies the IND-CPA property.

Theorem 4.9. *Let* $\mathrm{SC} = (\mathrm{enc}, \mathrm{dec})$ *be an encryption scheme and* $\mathbf{sc}$ *be the pair of converters described above. Let* ε_2 *be a performance measure such that with* $\varepsilon \coloneqq (0, \varepsilon_2)$*:*

$$\left(\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet, \langle\bullet\!\leftarrow\ \rightarrow\rangle\right) \quad \xrightarrow{\mathbf{sc},\sigma_{\text{CPA}},\varepsilon} \quad \langle\bullet\!\leftarrow\ \rightarrow\!\bullet\rangle\,,$$

where the symbols refer to sequences $\langle\bullet\!\leftarrow\ \rightarrow\rangle_{[q]}$ *and* $\langle\bullet\!\leftarrow\ \rightarrow\rangle_{[q]}$ *(and* σ_{CPA} *is as described above). Then the scheme* SC *is correct and*

$$[\![\mathbf{G}_q^{\text{IND-CPA}}(\mathrm{SC})]\!] \quad \leq \quad \varepsilon_2 \circ (\cdot\mathbf{C}_{\text{CPA}}^{-1}) \tag{4.1}$$

for a reduction system $\mathbf{C}_{\text{CPA}}^{-1}$ *described in the proof.*

Proof. The correctness follows from the fact that

$$\mathsf{enc}^A\mathsf{dec}^B\left(\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet, \langle\bullet\!\leftarrow\ \rightarrow\rangle_{[1]}\right) \quad \equiv \quad \langle\bullet\!\leftarrow\ \rightarrow\!\bullet_{\perp}\rangle_{[1]}$$

and from the definition of $\mathbf{sc}$, where a plaintext is encrypted and the resulting ciphertext is decrypted with respect to the same key, as in the correctness condition.

To show equation (4.1), we exhibit a reduction system $\mathbf{C}_{\text{CPA}}^{-1}$ that behaves as follows.[5] Upon a query $\mathbf{enc}(m)$ at the outside interface, output m at the inside A-sub-interface, and upon obtaining c as an input at the inside E-sub-interface, respond with c. Since

$$\mathbf{G}_{0,q}^{\text{IND-CPA}}(\mathrm{SC}) \quad \equiv \quad \mathbf{C}_{\text{CPA}}^{-1}\left(\mathsf{enc}^A\mathsf{dec}^B\left(\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet, \langle\bullet\!\leftarrow\ \rightarrow\rangle_{[q]}\right)\right)$$

and

$$\mathbf{G}_{1,q}^{\text{IND-CPA}}(\mathrm{SC}) \quad \equiv \quad \mathbf{C}_{\text{CPA}}^{-1}\left({\sigma_{\text{CPA}}}^E\,\langle\bullet\!\leftarrow\ \rightarrow\!\bullet\rangle_{[q]}\right),$$

the statement follows by Lemma 2.6. □

[5]The reduction system has to be described in the opposite direction compared to Definition 3.24.

Indistinguishability of Ciphertexts under Chosen-Ciphertext Attack—IND-CCA

The IND-CCA game is supposed to capture a confidentiality guarantee in a setting where the ciphertexts are not transmitted authentically, but over an insecure connection. In the game, the adversary is, in addition to one type of oracle of the IND-CPA game, given access to a decryption oracle where it can query ciphertexts that are different from those he obtained from the encryption oracle.[6] A scheme which satisfies the IND-CCA property can be used to construct a non-malleable confidential channel from a shared secret key and an insecure channel; hence, it achieves the construction

$$(\bullet\!\!=\!\!=\!\!\bullet, - \twoheadrightarrow) \quad \xrightarrow{\mathsf{sc}} \quad -\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet.$$

The construction is shown relative to the simulator σ_{CCA}, which initially samples a key $k \leftarrow\!\!\$\, \mathcal{K}$. Upon receiving the i-th message length $n_i \in \mathbb{N}$ at the inside interface, the simulator samples a string $m_i \leftarrow\!\!\$\, \{0,1\}^{n_i}$, encrypts it as $c_i \leftarrow\!\!\$\, \mathrm{enc}\,(k, m_i)$, and outputs c_i at the outside interface. Upon receiving a ciphertext c' at the outside interface, if $c' = c_i$ for some $i \in \mathbb{N}$, then output i at the inside interface. Otherwise, σ_{CCA} decrypts c' as $m' \leftarrow \mathrm{dec}\,(k, c')$ and outputs m' at the inside interface.

The above described simulator is *not* a monotone system in the sense of Definition 3.18. This is intrinsic to the (required) behavior of the simulator: upon receiving a ciphertext c' at the outside interface, it has to either decrypt c' and inject the obtained plaintext into the channel, or (if the ciphertext had been output before) provide as input the index of the message to which it corresponds. This behavior cannot be described as a monotone function. The statement in Theorem 4.10 is still sound if one extends the composition operation on monotone discrete systems to general discrete systems. Theorem 3.13, however, does not apply because a system algebra based on this general type of system is not connection-order invariant. This can be resolved by considering *sets* of discrete systems and using techniques such as those described by Brock [Bro83], as sketched in Chapter 6, but is not in the scope of this thesis.

Theorem 4.10. *Let* $\mathrm{SC} = (\mathrm{enc}, \mathrm{dec})$ *be an encryption scheme and* **sc** *be the pair of converters described above. For the simulator* σ_{CCA}*,*

$$\left(\bullet\overset{\mathcal{K}}{=\!\!=}\!\!\bullet, - \twoheadrightarrow\right) \quad \xRightarrow{\mathsf{sc},\sigma_{\text{CCA}},(0,\varepsilon)} \quad -\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet,$$

[6] The reason for the latter restriction is that if the adversary were allowed to decrypt the challenge, winning the game would become trivial.

where the symbols refer to sequences $\overset{q\text{ msgs}}{-\twoheadrightarrow}$ *and* $\overset{q\text{ msgs}}{-\diamond\twoheadrightarrow\bullet}$*, and*

$$\varepsilon_q = [\![\big(\mathbf{G}_{0,q}^{\text{IND-CCA}}(\mathrm{SC}) \mid \mathbf{G}_{1,q}^{\text{IND-CCA}}(\mathrm{SC}) \big)]\!] \circ (\cdot \mathbf{C}_{\text{CCA}})$$

for a reduction converter $\mathbf{C}_{\text{CCA}}$ *described in the proof.*

Proof. The availability condition means that for each $q \in \mathbb{N}$, the equation

$$\mathsf{enc}^A \mathsf{dec}^B \left(\overset{\mathcal{K}}{\bullet\!=\!=\!\bullet}, \overset{q\text{ msgs}}{-\twoheadrightarrow}_{\perp} \right) \quad \equiv \quad \overset{q\text{ msgs}}{-\diamond\twoheadrightarrow\bullet}_{\perp}$$

holds. This follows directly from the correctness of the encryption scheme according to Definition 2.18 and the fact that both systems stop after receiving q inputs at the A-interface.

For the security condition, we describe a converter $\mathbf{C}_{\text{CCA}}$ such that

$$\left(\mathsf{enc}^A \mathsf{dec}^B \left(\overset{\mathcal{K}}{\bullet\!=\!=\!\bullet}, \overset{q\text{ msgs}}{-\twoheadrightarrow} \right) \;\middle|\; \sigma_{\text{CCA}}{}^E \overset{q\text{ msgs}}{-\diamond\twoheadrightarrow\bullet} \right) \equiv \big(\mathbf{C}_{\text{CCA}} \mathbf{G}_{0,q}^{\text{IND-CCA}}(\mathrm{SC}) \mid \mathbf{C}_{\text{CCA}} \mathbf{G}_{1,q}^{\text{IND-CCA}}(\mathrm{SC}) \big).$$

This converter $\mathbf{C}_{\text{CCA}}$ has three outside sub-interfaces (labeled A, B, and E) and behaves as follows: Upon input the i-th message m_i at the A-sub-interface, $\mathbf{C}_{\text{CCA}}$ queries $\mathbf{enc}(m_i)$ at the inside interface, obtaining a ciphertext c_i which it outputs at the outside E-sub-interface. Upon input the i-th ciphertext c_i' at the E-sub-interface, if $c_i' = c_j$ for some $j \in \mathbb{N}$, then $\mathbf{C}_{\text{CCA}}$ outputs m_j at the outside B-sub-interface. Otherwise, $\mathbf{C}_{\text{CCA}}$ queries $\mathbf{dec}(c_i')$ at the inside interface, obtaining m_i' which it outputs at the outside B-sub-interface.

Since

$$\mathsf{enc}^A \mathsf{dec}^B \left(\overset{\mathcal{K}}{\bullet\!=\!=\!\bullet}, \overset{q\text{ msgs}}{-\twoheadrightarrow} \right) \quad \equiv \quad \mathbf{C}_{\text{CCA}} \mathbf{G}_{0,q}^{\text{IND-CCA}}(\mathrm{SC})$$

and

$$\sigma_{\text{CCA}}{}^E \overset{q\text{ msgs}}{-\diamond\twoheadrightarrow\bullet} \quad \equiv \quad \mathbf{C}_{\text{CCA}} \mathbf{G}_{1,q}^{\text{IND-CCA}}(\mathrm{SC}),$$

the statement follows by Lemma 2.6. □

Indeed, IND-CCA is "almost" equivalent to the construction described in Theorem 4.10; this is shown by Maurer et al. [MRT12]. While IND-CCA is considered the standard notion for confidentiality in settings where the adversary can modify ciphertexts, the exact formalized guarantees appear unnatural. This is explained further in the following paragraphs.

Artificial strictness. Several authors have noted that IND-CCA is artificially strict in the sense that the decryption oracle will decrypt any ciphertext except for the exact challenge ciphertext [ADR02, CKN03, Kra01, Kro99, Sho01]. Schemes that allow for "obvious" ciphertext modifications are not IND-CCA secure, the typical separating example being an (otherwise IND-CCA secure) encryption scheme where the encryption always appends a single bit to the ciphertext, and this bit is ignored during decryption. While this modification does not hurt the security guarantees in any meaningful way, the resulting scheme is not IND-CCA secure.

Canetti et al. [CKN03] analyze several variants of "replayable" CCA security.[7] In the games they describe, not only the exact challenge ciphertext is disallowed in decryption queries, but also "related" ciphertexts. Intuitively, this means that encryption schemes may allow certain modifications to ciphertexts that do not change the result of the decryption. In more detail, the notions considered by Canetti et al. [CKN03] are:

- IND-RCCA, or "replayable CCA": any ciphertext that decrypts to one of the plaintexts issued to the encryption oracle is disallowed;

- IND-sd-RCCA, or "secretly detectable RCCA": intuitively, the receiver can detect whether an adversarially generated ciphertext was generated as a "modification" of an honestly generated one, or whether it is "independent" of all honestly generated ones, these "modified" ciphertexts are disallowed;

- IND-pd-RCCA, or "publicly detectable RCCA": the above distinction can be done publicly, i.e., without knowledge of the secret key.

The exact formalization is technically involved; for details, we refer to the work of Canetti et al. [CKN03]. As discussed by Maurer et al. [MRT12], the constructive definition of encryption is closely related to IND-sd-RCCA, because the simulator (who knows the secret key) needs to identify the ciphertexts that relate to ciphertexts simulated before.

Unnatural Malleability. IND-CCA is not a natural security requirement for symmetric encryption: The adversary may generate valid ciphertexts for arbitrary plaintexts (but only independently of honestly sent messages). Most existing symmetric encryption schemes are either malleable (such as the one-time pad or CBC) or, if they are non-malleable, they already construct the

[7] Their original notions regard public-key schemes, but the extensions to symmetric schemes are also described.

fully secure channel (such as authenticated encryption). Here, it becomes apparent that IND-CCA has evolved as a notion for public-key schemes, where the adversary knows the encryption key and can encrypt arbitrary messages.

Integrity of Ciphertexts—INT-CTXT

Integrity of ciphertexts has first been introduced by Bellare and Namprempre [BN00] and formalizes that the adversary cannot produce *any* fresh valid ciphertext. In more detail, an encryption scheme is said to achieve INT-CTXT security if no adversary with access to an encryption oracle can generate a valid ciphertext that is different from all ciphertexts obtained from the oracle. Here, "valid" means that the decryption outputs a message (not an error symbol). Note that existential unforgeability [KY00b] and ciphertext unforgeability [Kra01] are similar: The differences are, for example, that the definition of Bellare and Namprempre[BN08] allows multiple queries to the challenge oracle, whereas the one of Katz and Yung [KY00b] allows only one.

A symmetric encryption scheme that satisfies both IND-CPA and INT-CTXT gives rise to a protocol that constructs a fully secure channel from an insecure channel, which can be written as

$$(\bullet\!\!=\ =\!\!\bullet,\ -\twoheadrightarrow) \quad \xrightarrow{\mathsf{sc}} \quad \bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\!\bullet.$$

Yet, INT-CTXT, similarly to IND-CCA, is artificially strict concerning modifications of ciphertexts and hence the converse statement does not hold.

We prove the construction statement relative to the following simulator σ_{CTXT}. The simulator initially samples a key $k \leftarrow\!\!\$\ \mathcal{K}$. Upon receiving the i-th message length $n_i \in \mathbb{N}$ at the inside interface, it samples a string $m_i \leftarrow\!\!\$\ \{0,1\}^{n_i}$, encrypts it as $c_i \leftarrow\!\!\$\ \mathrm{enc}\,(k, m_i)$, and outputs c_i at the outside interface. Upon receiving a ciphertext c' at the outside interface, if $c' = c_i$ for some $i \in \mathbb{N}$, then output i at the inside interface. Note that this simulator, in contrast to σ_{CCA}, is monotone.

Theorem 4.11. *Let* $\mathrm{SC} = (\mathrm{enc}, \mathrm{dec})$ *be an encryption scheme and* **sc** *be the pair of converters described above. For the simulator* σ_{CTXT}*,*

$$\left(\bullet\!\!\overset{\mathcal{K}}{=}\ =\!\!\bullet,\ -\twoheadrightarrow\right) \quad \xRightarrow{\mathsf{sc},\sigma_{\text{CTXT}},(0,\varepsilon)} \quad \bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow\!\!\bullet,$$

where the symbols refer to sequences $-\overset{q\ \text{msgs}}{\twoheadrightarrow}$ *and* $\bullet\!\!-\!\!\diamond\!\!-\overset{q\ \text{msgs}}{\twoheadrightarrow}\!\!\bullet$*, and*

$$\varepsilon_q = [\![(\mathbf{G}^{\text{IND-CPA}}_{0,q}(\mathrm{SC}) \mid \mathbf{G}^{\text{IND-CPA}}_{1,q}(\mathrm{SC}))]\!] \circ (\cdot\mathbf{C}''_{\text{CPA}}) + [\![\mathbf{G}^{\text{INT-CTXT}}_{q}(\mathrm{SC})]\!] \circ (\cdot\mathbf{C}_{\text{CTXT}})$$

for reduction converters $\mathbf{C}''_{\text{CPA}}$ *and* $\mathbf{C}_{\text{CTXT}}$ *described in the proof.*

Proof. The availability condition means that for each $q \in \mathbb{N}$, the equation

$$\mathsf{enc}^A \mathsf{dec}^B \left(\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet, \overset{q\ \text{msgs}}{-\!\!\twoheadrightarrow}\!\perp \right) \quad \equiv \quad \overset{q\ \text{msgs}}{\bullet\!-\!\diamond\!\twoheadrightarrow}\!\bullet\!\perp$$

holds. This follows directly from the correctness of the encryption scheme according to Definition 2.18 and because both systems stop after receiving exactly q inputs at the A-interface.

For the security condition, we bound the advantage in the distinction problem $(\mathbf{R}_q \mid \mathbf{S}_q)$, with

$$\mathbf{R}_q \quad \coloneqq \quad \mathsf{enc}^A \mathsf{dec}^B \left(\bullet\!\overset{\mathcal{K}}{=\!=}\!\bullet, \overset{q\ \text{msgs}}{-\!\!\twoheadrightarrow} \right) \qquad \text{and} \qquad \mathbf{S}_q \quad \coloneqq \quad \sigma_{\text{CTXT}}{}^E \overset{q\ \text{msgs}}{\bullet\!-\!\diamond\!\twoheadrightarrow}\!\bullet.$$

We first describe a converter $\mathbf{C}_{\text{CTXT}}$ that has three outside sub-interfaces (labeled A, B, and E) and behaves as follows: Upon input the i-th message m_i at the A-sub-interface, $\mathbf{C}_{\text{CTXT}}$ queries $\mathbf{enc}(m_i)$ at the inside interface, obtaining a ciphertext c_i which it outputs at the outside E-sub-interface. Upon input ciphertext c' at the E-sub-interface, if $c' = c_i$ for some $i \in \mathbb{N}$ then $\mathbf{C}_{\text{CTXT}}$ outputs m_i at the outside B-sub-interface. Otherwise, $\mathbf{C}_{\text{CTXT}}$ queries $\mathbf{check}(c'_i)$ at the inside interface (and does not provide output at the outside interface).

We define an MBO on the systems $\mathbf{R}_q$ that becomes 1 once an input at the E-interface is a ciphertext that was not output at the E-interface before but is valid according to the key in dec. The game equivalence

$$\mathbf{R}_q \quad \overset{\text{g}}{\equiv} \quad \mathbf{C}_{\text{CTXT}} \mathbf{G}_q^{\text{INT-CTXT}}(\text{SC})$$

holds because each input at the A-interface results in an output at the E-interface which is an encryption under a uniformly random key, and inputs at the E-interface result in outputs at the B-interface if and only if one of the previously obtained ciphertexts is input. Moreover, the MBO becomes 1 in both cases if a ciphertext is input at the E-interface that is valid but not output at the E-interface before. Lemma 2.14 then implies that

$$[\![(\mathbf{R}_q \mid \mathbf{C}_{\text{CTXT}} \mathbf{G}_q^{\text{INT-CTXT}}(\text{SC}))]\!] \quad \leq \quad [\![\mathbf{G}_q^{\text{INT-CTXT}}(\text{SC})]\!] \circ (\cdot \mathbf{C}_{\text{CTXT}}).$$

Our next step is to describe a reduction converter $\mathbf{C}''_{\text{CPA}}$ such that

$$\mathbf{C}''_{\text{CPA}} \mathbf{G}_{0,q}^{\text{IND-CPA}}(\text{SC}) \quad \equiv \quad \mathbf{C}_{\text{CTXT}} \mathbf{G}_q^{\text{INT-CTXT}}(\text{SC})$$

and

$$\mathbf{C}''_{\text{CPA}} \mathbf{G}_{1,q}^{\text{IND-CPA}}(\text{SC}) \quad \equiv \quad \mathbf{S}_q.$$

This converter $\mathbf{C}''_{\text{CPA}}$ has three outside sub-interfaces (labeled A, B, and E) and behaves as follows: Upon input the i-th message m_i at the A-sub-interface, $\mathbf{C}''_{\text{CPA}}$ queries $\mathbf{enc}(m_i)$ at the inside interface, obtaining a ciphertext c_i which it outputs at the outside E-sub-interface. Upon input the i-th ciphertext c'_i at the E-sub-interface, if $c'_i = c_j$ for some j then output m_j at the B-sub-interface. The equivalences

$$\mathbf{C}_{\text{CTXT}}\mathbf{G}_q^{\text{INT-CTXT}}(\text{SC}) \quad \equiv \quad \mathbf{C}''_{\text{CPA}}\mathbf{G}_{0,q}^{\text{IND-CPA}}(\text{SC})$$

and

$$\sigma_{\text{CTXT}}{}^{E}\bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet^{\,q\text{ msgs}} \quad \equiv \quad \mathbf{C}''_{\text{CPA}}\mathbf{G}_{1,q}^{\text{IND-CPA}}(\text{SC}),$$

hold by the definitions of the systems, so the statement follows by Lemma 2.6. The triangle inequality for the performance in the distinction problem allows us to conclude that

$$\begin{aligned}
&\left[\!\!\left[\left(\mathsf{enc}^A\mathsf{dec}^B\left(\bullet\!\overset{\mathcal{K}}{=}\!\bullet,\ -\overset{q\text{ msgs}}{\twoheadrightarrow}\right)\ \middle|\ \sigma_{\text{CTXT}}{}^{E}\bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet^{\,q\text{ msgs}}\right)\right]\!\!\right] \\
&\qquad \le \left[\!\!\left[\left(\mathsf{enc}^A\mathsf{dec}^B\left(\bullet\!\overset{\mathcal{K}}{=}\!\bullet,\ -\overset{q\text{ msgs}}{\twoheadrightarrow}\right)\ \middle|\ \mathbf{C}_{\text{CTXT}}\mathbf{G}_q^{\text{INT-CTXT}}(\text{SC})\right)\right]\!\!\right] \\
&\qquad\qquad + \left[\!\!\left[\left(\mathbf{C}_{\text{CTXT}}\mathbf{G}_q^{\text{INT-CTXT}}(\text{SC})\ \middle|\ \sigma_{\text{CTXT}}{}^{E}\bullet\!\!-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet^{\,q\text{ msgs}}\right)\right]\!\!\right] \\
&\le \left[\!\!\left[\mathbf{G}_q^{\text{INT-CTXT}}(\text{SC})\right]\!\!\right] \circ (\cdot\mathbf{C}_{\text{CTXT}}) + \left[\!\!\left[\left(\mathbf{G}_{0,q}^{\text{IND-CPA}}(\text{SC})\ \middle|\ \mathbf{G}_{1,q}^{\text{IND-CPA}}(\text{SC})\right)\right]\!\!\right] \circ (\cdot\mathbf{C}''_{\text{CPA}}),
\end{aligned}$$

which concludes the proof □

4.5 Combining Encryption and Authentication

A secure channel can be constructed via the application of a MAC (for authenticity) and an encryption scheme (for confidentiality). There are two *a priori* equally valid generic ways for combining the MAC and the encryption scheme. Either, one first uses a MAC to construct an authenticated channel from an insecure channel and a shared secret key, and then uses an encryption scheme to construct a fully secure channel. This is usually referred to as Encrypt-then-Authenticate due to the duality discussed in Section 1.2.1. Or, one first uses an encryption scheme to construct a confidential but potentially malleable channel from an insecure channel and a shared secret key, and then uses a MAC to construct a fully secure channel. This is usually referred to as Authenticate-then-Encrypt.

4.5.1 Encrypt-then-Authenticate

The statement about Encrypt-then-Authenticate follows as a simple corollary of the statements proven in Sections 4.3 and 4.4. For the single-message case,

we obtain as a direct corollary of Theorem 4.2 that

$$(\bullet\!= =\!\bullet, - \rightarrow) \quad \xRightarrow{\mathsf{mac},\sigma_{\mathsf{mac}},\varepsilon_{\mathsf{mac}}} \quad \bullet\!- \rightarrow,$$

with the simulator σ_{mac} and the performance measure $\varepsilon_{\mathsf{mac}}$ described in the theorem. Also, as a corollary of Theorem 4.7, we obtain that

$$(\bullet\!= =\!\bullet, \langle \bullet\!- \rightarrow \rangle) \quad \xRightarrow{\mathsf{sc},\sigma_{\mathrm{CPA}},\varepsilon_{\mathrm{CPA}}} \quad \langle \bullet\!- \rightarrow\!\bullet \rangle,$$

with the simulator σ_{CPA} and the performance measure $\varepsilon_{\mathrm{CPA}}$ described in the theorem.

Using the composition theorem (Theorem 3.13), we directly obtain

$$(\bullet\!= =\!\bullet, (\bullet\!= =\!\bullet, - \rightarrow)) \quad \xRightarrow{\mathsf{sc}\circ\langle\mathsf{mac}\rangle_2,\langle\sigma_{\mathsf{mac}}\rangle_2\circ\sigma_{\mathrm{CPA}},\varepsilon} \quad \bullet\!- \rightarrow\!\bullet,$$

with $\varepsilon = (0, \varepsilon_{\mathrm{CPA}} \circ (\cdot \langle\sigma_{\mathsf{mac}}\rangle_2) + \varepsilon_{\mathsf{mac}} \circ (\cdot\mathsf{sc}))$. The analogous statements hold of course with respect to the one-time pad encryption and the CBC-mode encryption described in Sections 4.4.1 and 4.4.2.

If both the MAC and the encryption apply to a multi-message case, the same holds for the composite construction. Hence, as a corollary to Theorems 4.2 and 4.3 and Corollary 4.8, we obtain

$$(\bullet\!= =\!\bullet, (\bullet\!= =\!\bullet, - \twoheadrightarrow)) \quad \xRightarrow{\mathsf{sc}\circ\mathsf{seq}\circ\langle\mathsf{mac}\rangle_2,\langle\sigma_{\mathsf{mac}}\rangle_2\circ\sigma_{\mathsf{seq}}\circ\sigma'_{\mathrm{CPA}},(0,\varepsilon)} \quad \bullet\!- \twoheadrightarrow\!\bullet,$$

where the symbols are parametrized in $q \in \mathbb{N}$ as $\overset{q\ \mathrm{msgs}}{-\twoheadrightarrow}$ and $\overset{q\ \mathrm{msgs}}{\bullet\!- \twoheadrightarrow\!\bullet}$ and with $\varepsilon_q = \varepsilon_{\mathrm{CPA},q} \circ (\cdot\sigma_{\mathsf{seq}}) \circ (\cdot \langle\sigma_{\mathsf{mac}}\rangle_2) + \varepsilon_{\mathsf{mac},q} \circ (\cdot\mathsf{seq}) \circ (\cdot\mathsf{sc})$.

Similar soundness results for Encrypt-then-Authenticate have been shown in different formal contexts (and often without an explicit notion of sequence numbers), for instance by Bellare and Namprempre [BN08].

4.5.2 Authenticate-then-Encrypt

In contrast to Encrypt-then-Authenticate, Authenticate-then-Encrypt is not generically secure. This can be seen by considering the following scheme, which was described by Maurer and Tackmann [MT10] and is slightly modified from Krawczyk's example [Kra01]: A message $m \in \{0,1\}^\ell$ is encoded bit-wisely to a message $m' \in \{0,1\}^{2\ell}$, before m' is encrypted with a one-time pad to guarantee confidentiality. The encoding of bits, however, is asymmetric: A 0-bit is encoded to 00, while a 1-bit is encoded to either $10, 01,$ or 11. Hence, if the attacker flips two subsequent bits c_{2i}, c_{2i+1} of the ciphertext, the corresponding plaintext bit m_i will always flip if $m_i = 0$, but flips with probability only $\frac{1}{3}$ if $m_i = 1$. As the verification of a strongly unforgeable

MAC will succeed if and only if the authenticated plaintext is unchanged, an attacker can guess the bit m_i with substantial probability by flipping c_{2i}, c_{2i+1} and checking whether the verification succeeds. Hence, the verification of the MAC leaks the bit at the corresponding position, which breaks confidentiality.

More abstractly, if the malleability allows the attacker to transform the ciphertext such that the probability of the plaintext to remain constant differs substantially for two different values of the plaintext, then the attacker can use the result of the MAC verification (valid or invalid) to detect which one of the two plaintexts was sent. For particular encryption schemes which do not exhibit the described weakness, the combination with a strongly unforgeable MAC is indeed secure. The first formal such statement has been shown by Krawczyk [Kra01]. Subsequently to our treatment [MT10], Paterson et al. [PRS11] have provided a game-based analysis of the schemes used in TLS which also applies to the variable-length padding applied there.

We show the soundness of Authenticate-then-Encrypt for the single-use XOR-malleable channel $\multimap\!\!\oplus\!\!\to\!\bullet$ here; a treatment which is specific to the exact combinations of schemes used in TLS appears in Section 5.

Let $\mathbf{mac} = (\mathsf{tag}, \mathsf{chk})$ be a pair of converters derived from a MAC scheme $\mathrm{MAC} = (\mathrm{mac}, \mathrm{check})$ with tag length $\tau \in \mathbb{N}$ as described in Section 4.3. We use the following simulator $\sigma'_{\mathbf{mac}}$: Upon input a message length $n \in \mathbb{N}$ at the inside interface, output a uniformly random string $c \leftarrow\!\!\$\ \{0,1\}^{n+t}$ of length $n+\tau$ at the outside interface. Upon input a ciphertext c' at the outside interface, if $c = c'$ then provide the trigger at the inside interface.

Theorem 4.12. *Let* $(\mathrm{mac}, \mathrm{check})$ *be a MAC scheme and* $\mathbf{mac} = (\mathsf{tag}, \mathsf{chk})$ *be the protocol described above. For the simulator* $\sigma'_{\mathbf{mac}}$,

$$\left(\bullet\!\!\overset{\mathcal{K}}{\leftarrow}\!\!\Rightarrow\!\!\bullet,\ \multimap\!\!\oplus\!\!\to\!\bullet\right) \xRightarrow{\mathbf{mac},\sigma'_{\mathbf{mac}},(0,\varepsilon)} \bullet\!\!\leftarrow\!\!\to\!\bullet,$$

where the statement is parametrized in $\ell \in \mathbb{N}$ *for the resources* $\overset{\ell+\tau\text{ bits}}{\multimap\!\!\oplus\!\!\to}\!\bullet$ *and* $\bullet\!\!\overset{\ell\text{ bits}}{\leftarrow\!\!\to}\!\bullet$, *and performance mappings* $\varepsilon \coloneqq [\![\mathbf{G}_1^{\text{SUF-CMA}}(\mathrm{MAC})]\!] \circ (\cdot\mathbf{C}_{\mathbf{mac}})'$, *for a reduction* $\mathbf{C}'_{\mathbf{mac}}$ *explicitly described in the proof.*

Proof. For the availability condition, we show

$$\mathsf{tag}^A\mathsf{chk}^B\left(\bullet\!\!\leftarrow\!\!\Rightarrow\!\!\bullet_\perp,\ \overset{\ell+\tau\text{ bits}}{\multimap\!\!\oplus\!\!\to}\!\bullet_\perp\right) \quad\equiv\quad \bullet\!\!\overset{\ell\text{ bits}}{\leftarrow\!\!\to}\!\bullet_\perp.$$

This follows directly from the correctness of the MAC scheme: for the key $k \in \mathcal{K}$ output by $\bullet\!\!\leftarrow\!\!\Rightarrow\!\!\bullet$ and all messages $m \in \mathcal{M}$, $\mathrm{check}(k, m, \mathrm{mac}(k, m)) = \underline{\mathsf{true}}$ and hence chk accepts the tag generated by tag. In both cases, the system provides output unless a message of length greater than ℓ messages have been input.

For the security condition, we define

$$\mathbf{R}_\ell \coloneqq \mathsf{tag}^A\mathsf{chk}^B\left(\bullet\!\!=\!\!=\!\!\bullet,\ \xrightarrow{\ell+\tau\text{ bits}}\!\!\!\!\oplus\!\!\!\!\rightarrow\bullet\right) \quad \text{and} \quad \mathbf{S}_\ell \coloneqq \sigma'_{\mathsf{mac}}{}^E\bullet\!\!\xleftarrow{\ell\text{ bits}}\!\!\!\!\rightarrow\bullet$$

and a reduction $\mathbf{C}'_{\mathsf{mac}}$ which we describe as follows. At the left interface, $\mathbf{C}'_{\mathsf{mac}}$ provides three sub-interfaces, while at the right interface it connects to $\mathbf{G}_1^{\text{SUF-CMA}}(\text{MAC})$. On input a message $m \in \mathcal{M}$ at the A-sub-interface, query it at the right interface to obtain a tag $t \in \{0,1\}^\tau$ and output a uniformly random string c of length $|m| + \tau$ at the E-sub-interface. On input a string c' at the E-sub-interface, continue as follows:

- If $|c| \geq |c'| \geq \tau$, then compute $m' = (m.t) \oplus c \oplus c'$ (the first $|c'|$ bits).
- If $|c| < |c'|$, then sample $s \leftarrow\!\!\$\ \{0,1\}^{|c'|-|c|}$ and compute $m' = ((m.t) \oplus c \oplus c').s$.

Let $m''.t' \leftarrow m'$ (i.e., the first $|m'| - \tau$ bits are assigned to m'', the remaining τ bits to t'). Query $\mathbf{vrf}\,(m'', t')$ at the right interface, if the game returns <u>true</u>, then output m at the B-sub-interface.

We then define an MBO on the systems $\mathbf{R}_q$ and $\mathbf{S}_q$ that becomes 1 if a message $c' \neq c$ is input at the E-interface but the MAC verification succeeds (within chk or σ'_{mac}, respectively); it is easy to see that

$$\hat{\mathbf{R}_{\mathbf{q}}} \quad \overset{\text{g}}{\equiv} \quad \hat{\mathbf{S}_{\mathbf{q}}} \quad \overset{\text{g}}{\equiv} \quad \mathbf{C}'_{\mathsf{mac}}\mathbf{G}_1^{\text{SUF-CMA}}(\text{MAC}).$$

By Lemma 2.14, we obtain that

$$[\![(\mathbf{R}_q \mid \mathbf{S}_q)]\!] \ \leq\ [\![\mathbf{C}'_{\mathsf{mac}}\mathbf{G}_1^{\text{SUF-CMA}}(\text{MAC})]\!] \ =\ [\![\mathbf{G}_1^{\text{SUF-CMA}}(\text{MAC})]\!] \circ (\cdot\mathbf{C}'_{\mathsf{mac}}),$$

and Lemma 2.6 gives us the desired result. □

4.5.3 Discussion

As discussed in the previous sections, both Encrypt-then-Authenticate and Authenticate-then-Encrypt can in principle be used to construct a fully secure channel from an insecure channel and two shared secret keys. The proof of Encrypt-then-Authenticate is generic in the sense that all MAC schemes are shown to construct an authenticated channel of the same type and the proof of the encryption schemes can assume this channel, moreover, the security definitions for the schemes correspond to the widely used notions WUF-CMA (for the MAC schemes) and IND-CPA (for the encryption schemes).

By contrast, the proof of Authenticate-then-Encrypt is not as generic. Each encryption scheme achieves a different type of malleability in the constructed

confidential channel, and the MAC has to be shown to be sufficient for each particular type of malleability. While one can characterize classes of malleability that are indeed sufficient (this approach is pursued in a previous paper [MT10]) and hence reduce the number of required proofs, this approach is still more fragile because, for instance, the particular encoding used for the pair of message and tag is important for the scheme to be secure (see the example in Section 4.5.2).

A third approach is to not use generic composition and directly show that a combined scheme achieves the construction in a single step. This approach, usually called *authenticated encryption*, is sufficient if one uses the (widely used) combination of proving the scheme to achieve both IND-CPA and INT-CTXT (this is shown in Theorem 4.11). The Encrypt-and-Authenticate (E&A) combination of a symmetric encryption scheme and a MAC, where the MAC is also computed on the plaintext, but the generated tag is not encrypted, cannot be described as a composition of two generic construction steps. It is an authenticated encryption under assumptions that are stronger than those for, e.g., EtA. Additionally, several monolithic *authenticated encryption* schemes that perform the complete transformation in a single step are described in the literature (e.g., [RBB03, WHF02]).

4.6 Receiver-Anonymous Communication

The main goal of this section is to model and achieve confidential and receiver-anonymous communication. We first formalize a useful anonymity guarantee in the resource $\Rightarrow\!\langle ? \rangle\!\!\Rightarrow\!\bullet$ in Section 4.6.1. We then discuss in Section 4.6.2 in which way this resource can be constructed by vanilla public-key encryption from a "broadcast" channel and several authenticated channels (to transmit the public keys). Finally, in Section 4.6.3 we show how to achieve this construction more efficiently, by means of a public-key encryption scheme with the properties IND-CCA, IK-CCA [BBDP01], and WROB-CCA [ABN10].

4.6.1 Resources for Receiver-Anonymous Communication

An n-receiver channel is a resource with an interface labeled A for the sender, n interfaces labeled $B_1, \ldots, B_n$ for the receivers, and a third type of interface labeled E that captures potential adversarial access. The security properties of different n-receiver channels are described in the following; the symbolic notation for the channels extends the one from Section 4.1.

Insecure broadcast communication. We base our constructions on a resource $- \preceq$, which allows the sender to broadcast a given message to all receivers $B_1, \ldots, B_n$. Such a channel can be implemented, for example, by multi-sending the same message individually to each receiver over an insecure network; the channel models also what is achieved by wireless broadcast. The resource $- \preceq$ leaks the complete message at the E-interface, and allows to delete, change, or inject messages destined for particular receivers via the E-interface. The behavior for the case where no attacker is present is described in System 26.

System 26 The insecure broadcast channel $- \preceq_{\perp}$

1: **once** $\forall j \leq i : A.j? \neq \square$
2: **for** $k \in [n]$ **do**
3: $B_k.i! \leftarrow A.i?$
4: **end for**
5: **end.**

If an attacker is present, the resource allows to control the entire communication via the E-interface. On input a message m at the A-interface, this message is leaked at the E-interface. The E-interface allows a potential attacker to target messages to the individual recipients, such as message m to receiver i or message $\tilde{m}$ to receiver i'. This is formally described as System 27 and depicted in Figure 4.6. At the E-interface, the channel has output sub-interfaces $E.1!, E.2!, \ldots$ which leak the messages input at the A-interface. The channel has, for each $k \in [n]$, input sub-interfaces $E.k.1?, E.k.2?, \ldots$, and a message input at $E.k.i?$ is output to the k-th receiver at $B_k.i!$.

System 27 The insecure broadcast channel $- \preceq$

1: **once** $\forall j \leq i : A.j? \neq \square$
2: $E.i! \leftarrow A.i?$
3: **end.**

4: **once** $\forall j \leq i : E.k.j? \neq \square$
5: $B_k.i! \leftarrow E.k.i?$
6: **end.**

Confidential receiver-anonymous communication. The confidential receiver-anonymous channel $\Rightarrow\!\!\!\diamond\!\!\!\Rightarrow$ leaks neither the message contents nor the intended recipient to the adversary, just the message length. It allows, however, to "conditionally" deliver a message to a chosen user if and only if this

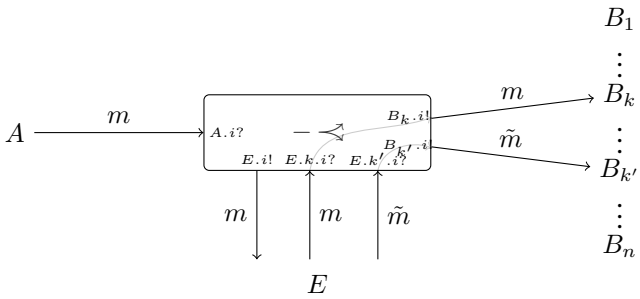


Figure 4.6: The insecure broadcast channel.

chosen user was the originally intended recipient. The behavior for the case where no attacker is present is described as System 28

System 28 The receiver-anonymous confidential channel $\rightleftarrows_{\perp}$

1: $\mathrm{cnt}_1, \ldots, \mathrm{cnt}_n \leftarrow 1$

2: **once** $\forall j \leq i : A.j? \neq \square$
3: $\quad (k, m) \leftarrow A.i?$
4: $\quad B_k.\mathrm{cnt}_k! \leftarrow m$
5: $\quad \mathrm{cnt}_k \leftarrow \mathrm{cnt}_k + 1$
6: **end.**

The behavior for the case where an attacker is present, including the capability to conditionally deliver messages, is described as System 29. The E-interface has sub-interfaces which are labeled by pairs $(l, i) \in [n] \times \mathbb{N}$—providing input at sub-interface $E.l.i?$ corresponds to injecting or delivering the i-th message to receiver B_l. The resource is also depicted in Figure 4.7. In the application of a public-key cryptosystem to a broadcast network such as $- \rightsquigarrow$, the capabilities at the E-interface correspond to trial deliveries of intercepted messages and to adversarial encryptions.

Authenticated channels. Each receiver uses one authenticated channel to send its public key to the sender. We model the composition of n such channels as a combined resource which has n sub-interfaces at both the A and the E interface and one interface for each B_i, $i \in [n]$. The channel can be seen as n copies of the channel $\leftarrow \bullet$ as described in Section 4.1.3, where for each $i \in [n]$ there is one channel from interface B_i to sub-interface $A.i$ and with adversarial sub-interface $E.i$. We denote this resource as $\prod_{(I,J)\in\mathcal{P}} \leftarrow \bullet$

System 29 The receiver-anonymous confidential channel $\Rightarrow\!\!\!\!\diamond\!\!\!\Rightarrow\!\!\!\bullet_{\mathcal{I}_\mathsf{C}}$ with corruption set $\mathcal{I}_\mathsf{C}$

$\mathrm{cnt}_1, \ldots, \mathrm{cnt}_n \leftarrow 1$

once $\forall j \leq i : A.j? \neq \square$
  $(k, m) \leftarrow A.i?$
  if $k \in \mathcal{I}_\mathsf{C}$ **then**
    $E.i! \leftarrow m$ ▷ Receiver corrupt: attacker learns the message.
  else
    $E.i! \leftarrow |m|$ ▷ Receiver honest: attacker learns the message length.
  end if
end.

once $\forall j \leq i : E.l.j? \neq \square$
  if $E.l.j? = i \in \mathbb{N}$ **then** ▷ Deliver a message to B_l.
    $(k, m) \leftarrow A.i?$
    if $k = l$ **then** ▷ Is B_l intended recipient?
      $B_l.\mathrm{cnt}_l! \leftarrow m$
      $\mathrm{cnt}_l \leftarrow \mathrm{cnt}_l + 1$
    end if
  else ▷ Inject a message to B_l.
    $B_l.\mathrm{cnt}_l! \leftarrow E.l.j?$
    $\mathrm{cnt}_l \leftarrow \mathrm{cnt}_l + 1$
  end if
end.

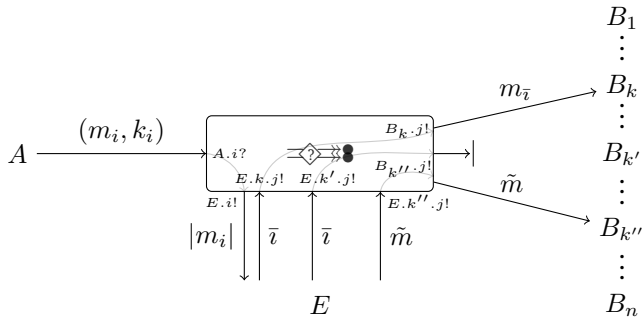


Figure 4.7: The confidential receiver-anonymous channel.

with the set $\mathcal{P} := \{(A.i, B_i) : i \in [n]\}$ indicating the (sub-)interface names. This is to be understood as the parallel composition of one resource $\leftarrow\!\bullet$ per $i \in [n]$, where the A-interfaces of the channel $\leftarrow\!\bullet$ (which has interfaces A, B, and E) is relabeled as $A.i$-sub-interface and the B-interfaces are relabeled as B_i-interface. For brevity, we will also use the notation $\leftarrow\!\bullet^n$.

4.6.2 Generic Construction using Public-Key Encryption

The application of public-key encryption in a setting with only one sender and one receiver can be described as deploying converters[8] pkenc_1 (associated with the sender) and pkdec_1 (associated with the receiver) as explained by Coretti et al. [CMT13a]. The receiver (within pkdec_1) initially uses the key-generation function PKEgen to obtain a key pair (sk, pk), stores the private key sk locally, and sends the public key pk via an authenticated channel (denoted $\leftarrow\!\bullet$, the first assumed resource). Upon receiving a ciphertext $\tilde{c}$ at the inside interface (via an insecure communication channel $-\!\rightarrow$, the second assumed resource), pkdec_1 computes $\tilde{m} \leftarrow \mathrm{PKEdec}(sk, \tilde{c})$ and outputs $\tilde{m}$ if $\tilde{m} \neq \diamond$. The encryption converter pkenc_1 initially obtains the public key pk (via $\leftarrow\!\bullet$) and, for each message m obtained at the outside interface, pkenc_1 computes $c \leftarrow\!\!\$\ \mathrm{PKEenc}(pk, m)$ and sends c over the insecure channel $-\!\rightarrow$. As pointed out already by Maurer and Schmid [MS96] and formally proved by Coretti et al. [CMT13a], this constructs a confidential channel $-\!\rightarrow\!\bullet$.

The receiver-anonymous channel $\Rightarrow\!\!\!\!\!\!\bullet$ can be constructed from $-\!\prec\!\!\!\!\!\!\triangleleft$ and one channel $\leftarrow\!\bullet$ per receiver using any secure public-key scheme: Each receiver generates a key pair and sends the public key through its authenti-

[8] We deviate from the notation of Coretti et al. [CMT13a] to differentiate between converters corresponding to symmetric encryption and converters corresponding to public-key encryption. The notation we use here is adapted from the one used by Kohlweiss et al. [KMO+13].

cated channel $\leftarrow \bullet$ to the sender; the sender transmits a message to a specific receiver by concatenating (in a fixed predetermined order): an encryption of this message under the intended receiver's public key and a "garbage" message encrypted with the appropriate key for each additional potential receiver; this composite message is then sent via the broadcast channel. Each receiver decrypts only "its" part of the composite ciphertext and checks whether or not the message was "garbage." (A simple way to implement the above "garbage" message is to set it to a public constant message $\bar{m} \in \mathcal{M}$ which is used only for that purpose.) If the broadcast channel is achieved by multi-sending the same message to each receiver, then one can also send only the corresponding part to each receiver.

Yet, this approach has two main disadvantages. First, the computation and communication complexity is linear in the (potentially large) number of possible receivers. Second, the sender must *know* the public keys of all potential receivers, not just of the one intended receiver.

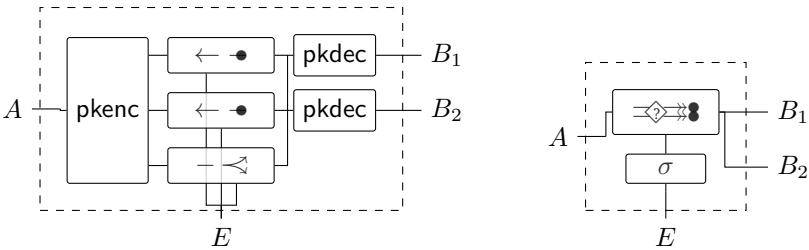
4.6.3 Achieving Confidential Receiver-Anonymous Communication

In this section, we consider PKE schemes deployed in a setting with one sender A and n receivers $B_1, \ldots, B_n$. A protocol is formally a tuple with one converter pkenc for A and one converter pkdec for each B_i with $i \in [n]$. Each converter pkdec is defined similarly to pkdec_1 in Section 4.6.2. The encryption converter pkenc connects at its inside interface to two resources. By using the n sub-interfaces of the first resource (each sub-interface corresponds to one authenticated channel for each receiver B_i), pkenc expects to obtain public keys $pk_1, \ldots, pk_n$. Upon receiving $(m, i) \in \mathcal{M} \times [n]$ at the outside interface, pkenc computes $c \leftarrow\$ \mathrm{PKEenc}(pk_i, m)$ and sends c via the second resource (instantiated by an insecure broadcast network $- \prec$) at the inside interface.

A public-key encryption scheme constructs the resource $\Rightarrow$ from a broadcast channel if it has the properties IND-CCA, IK-CCA, and WROB-CCA. The property WROB-CCA (weak robustness) captures the guarantee that ciphertexts honestly generated for one user will not be successfully decrypted by another user. We show that weak robustness is sufficient for our construction; in view of the results of Abdalla et al. [ABN10], this may be surprising, since the adversary can inject arbitrary ciphertexts into the channel $- \prec$. The intuitive reason why WROB-CCA is sufficient is two-fold: First, preventing the adversary from generating a single "fresh" ciphertext that is accepted by two receivers is only helpful if, for some reason, injecting two different ciphertexts is impossible, or more difficult for the adversary than injecting a single one. Second, the non-malleability guarantees of IND-CCA exclude that the

adversary can "maul" honestly generated ciphertexts such that unintended receivers decrypt "related" plaintexts (this is used in the reduction to IND-CCA in the proof of Theorem 4.13).

In a setting with multiple receivers it becomes relevant to capture the guarantees that are provided in case one or more of the receivers are corrupted. We only consider static corruptions here which can be captured by considering, for each subset $\mathcal{I}_{\mathrm{C}} \subseteq [n]$, a specific construction statement in which, intuitively, the capabilities of the interfaces B_i for $i \in \mathcal{I}_{\mathrm{C}}$ are also provided at the E-interface, as discussed in Section 3.3.3.



(a) Using public-key encryption over an (insecure) broadcast.

(b) Communication via the anonymous confidential network.

Figure 4.8: The security statement in a setting with two receivers.

The security statement we prove below is depicted in Figure 4.8, where we show how the scheme is used together with the assumed resources: Each sender transmits its public key authentically to the sender, who then uses the broadcast channel to transmit the ciphertext to both receivers. Figure 4.8b shows the idealized setting, where the message is transmitted via the resource (which guarantees confidentiality). The value "$*$" is determined by the simulator and depends on the values $\tilde{c}_1$ and $\tilde{c}_2$ given by the adversary; the symbol may stand for a query to deliver the message m or to inject unrelated messages.

Theorem 4.13 shows that if the public-key encryption scheme has the three assumed properties, then the two settings in Figure 4.8 are indistinguishable. The intuitive interpretation of this statement is that whenever such a scheme is used to protect messages transmitted via a broadcast channel such as $-\!\!\prec$, one obtains the guarantees explicitly described by the "idealized" network resource . The proof of the theorem shows that every distinguisher for the two settings can be transformed into an adversary against (at least) one of the three properties IND-CCA, IK-CCA, and WROB-CCA with loss qn for q messages and n receivers.

We first describe, for each case $\mathcal{I}_{\mathrm{C}} \subseteq [n]$, the simulator $\sigma_{\mathcal{I}_{\mathrm{C}}}$ as follows. It

keeps internally (analogously to the channels) counters $\mathrm{cnt}_1, \ldots, \mathrm{cnt}_n \in \mathbb{N}$ which are initialized to 1.

- Generate $n - |\mathcal{I}_{\mathrm{C}}|$ private-/public-key pairs (pk_i, sk_i) with $i \in [n] \setminus \mathcal{I}_{\mathrm{C}}$. Output each pk_i at the outside sub-interface corresponding to the respective receiver at $\leftarrow\!\!\bullet^n$. Furthermore, generate one auxiliary key pair $(\tilde{pk}, \tilde{sk})$. For all $j \in \mathcal{I}_{\mathrm{C}}$, obtain a public key $\bar{pk}_i$ at the corresponding outside sub-interface of $\leftarrow\!\!\bullet$.

- Upon the k-th message length ℓ_k from $\Rrightarrow\!\!\langle ?\rangle\!\!\Rrightarrow\!\!\bullet$, compute a ciphertext $c_k \leftarrow\!\!\$\ \mathrm{PKEenc}(\tilde{pk}, 0^{\ell_k})$ and output c_k at the first outside sub-interface (corresponding to $-\!\!\lessdot$). Upon receiving a pair (m_k, i_k) at the inside interface (which means $i_k \in \mathcal{I}_{\mathrm{C}}$), encrypt m_i using $\bar{pk}_{i_k}$ and output that ciphertext at the first outside sub-interface.

- Upon input a message $\tilde{c}$ to some user $j \in [n] \setminus \mathcal{I}_{\mathrm{C}}$ at the outside sub-interface corresponding to $-\!\!\lessdot$:

  – In case $\tilde{c} = c_{\bar{k}}$ for some $\bar{k} \in \mathbb{N}$, output $\bar{k}$ at the inside interface $\mathsf{In}.j.\mathrm{cnt}_j?$ and set $\mathrm{cnt}_j \leftarrow \mathrm{cnt}_j + 1$.
  – In case $\tilde{c}$ is "fresh," compute $\tilde{m}_j \leftarrow \mathrm{PKEdec}(sk_j, \tilde{c})$, and, if $\tilde{m}_j \neq \diamond$, output $\tilde{m}_j$ at the inside interface $\mathsf{In}.j.\mathrm{cnt}_j?$ and set $\mathrm{cnt}_j \leftarrow \mathrm{cnt}_j + 1$.

The simulator described here is not monotone for the same reason as the simulator σ_{CCA} in Section 4.4.3. Analogously, the statement of Theorem 4.13 is still sound, but the composition theorem stated as Theorem 3.13 does not immediately apply. We remark, however, that the simulator (as σ_{CCA}) is of a type for which modularity can be proven using techniques such as those described by Brock [Bro83].

Theorem 4.13. *Let* PKE $=$ (PKEgen, PKEenc, PKEdec) *be a public-key encryption scheme and* pke $=$ (pkenc, pkdec, … pkdec) *be the protocol as described above. Then* pke *constructs* $\Rrightarrow\!\!\langle ?\rangle\!\!\Rrightarrow\!\!\bullet$ *from* $-\!\!\lessdot$ *and* $\prod_{(I,J)\in\mathcal{P}} \leftarrow\!\!\bullet$ *with respect to static corruptions. More formally, for each* $\mathcal{I}_{\mathrm{C}} \subseteq [n]$,

$$\left(-\!\!\lessdot_{\mathcal{I}_{\mathrm{C}}}, \leftarrow\!\!\bullet^n_{\mathcal{I}_{\mathrm{C}}}\right) \quad \xmapsto{\mathsf{pke}, \sigma_{\mathcal{I}_{\mathrm{C}}}, \varepsilon} \quad \Rrightarrow\!\!\langle ?\rangle\!\!\Rrightarrow\!\!\bullet_{\mathcal{I}_{\mathrm{C}}},$$

where the symbols correspond to the families $\leftarrow\!\!\bullet^n_{\mathcal{I}_{\mathrm{C}}}$, $\overset{q\ \mathrm{msgs}}{-\!\!\lessdot}_{\mathcal{I}_{\mathrm{C}}}$ *and* $\overset{q\ \mathrm{msgs}}{\Rrightarrow\!\!\langle ?\rangle\!\!\Rrightarrow\!\!\bullet}_{\mathcal{I}_{\mathrm{C}}}$ *with parameters* $q, n \in \mathbb{N}$ *and* $\mathcal{I}_{\mathrm{C}} \subseteq \mathcal{I}$, *and*

$$\varepsilon^1_q \coloneqq \left[\!\left[\mathbf{G}^{\text{WROB-CCA}}_{n,q}(\mathrm{PKE})\right]\!\right] \circ (\cdot\mathbf{C}),$$

respectively

$$\varepsilon^2_q \coloneqq qn \cdot \left[\!\left[\left(\mathbf{G}^{\text{PKE-CCA}}_{q,0}(\mathrm{PKE}) \;\middle|\; \mathbf{G}^{\text{PKE-CCA}}_{q,1}(\mathrm{PKE})\right)\right]\!\right] \circ (\cdot\mathbf{C}')$$

$$+ qn \cdot [\![\big(\mathbf{G}^{\text{IK-CCA}}_{0,q}(\mathrm{PKE}) \mid \mathbf{G}^{\text{IK-CCA}}_{1,q}(\mathrm{PKE}) \big)]\!] \circ (\cdot \mathbf{C}'')$$
$$+ q \cdot [\![\mathbf{G}^{\text{WROB-CCA}}_{n,q}(\mathrm{PKE})]\!] \circ (\cdot \mathbf{C}'''),$$

for four reductions $\mathbf{C}$*,* $\mathbf{C}'$*,* $\mathbf{C}''$*, and* $\mathbf{C}'''$ *described in the proof.*

Proof. The availability condition means that for all $n, q \in \mathbb{N}$, the statement

$$[\![\Big(\mathsf{pkenc}^A \mathsf{pkdec}^{B_1} \cdots \mathsf{pkdec}^{B_n} \Big(\overset{q \text{ msgs}}{-\!\!\prec}_{\perp}, \leftarrow \bullet^n{}_{\perp} \Big) \mid \overset{q \text{ msgs}}{\Rightarrow\!\!\diamond\!\!?\!\!\Rightarrow\!\!\bullet\!\!\bullet}_{\perp} \Big)]\!] \leq q \, [\![\mathbf{G}^{\text{WROB-CCA}}_{n,q}(\mathrm{PKE})]\!] \circ (\cdot \mathbf{C})$$

holds, for a reduction $\mathbf{C}$ described as follows. Initially, $\mathbf{C}$ calls $\mathbf{genuser}(i)$ for $i \in [n]$ at the inside interface to obtain the public keys $pk_1, \ldots, pk_n$. Then, upon each input (k, m) at the A-sub-interface of the outside interface, $\mathbf{C}$ calls $\mathbf{enc}\,(m, k)$ at the inside interface and outputs m at the outside B_k sub-interface. We then use an MBO on

$$\mathbf{R}_{\perp} \quad \coloneqq \quad \mathsf{pkenc}^A \mathsf{pkdec}^{B_1} \cdots \mathsf{pkdec}^{B_n} \Big(\overset{q \text{ msgs}}{-\!\!\prec}_{\perp}, \leftarrow \bullet^n{}_{\perp} \Big)$$

that becomes 1 as soon as an input (k, m) at the A-interface leads to an output at some interface $B_{k'}$ for $k' \neq k$. If the encryption scheme has perfect correctness (if the scheme is *not* perfectly correct, we alter the MBO to take this into account), then we obtain

$$\hat{\mathbf{R}}_{\perp} \quad \overset{g}{\equiv} \quad \hat{\mathbf{S}}_{\perp} \quad \overset{g}{\equiv} \quad \mathbf{CG}^{\text{WROB-CCA}}_{n,q}(\mathrm{PKE}),$$

(where $\mathbf{S}_{\perp} \coloneqq \overset{q \text{ msgs}}{\Rightarrow\!\!\diamond\!\!?\!\!\Rightarrow\!\!\bullet\!\!\bullet}_{\perp}$) and hence by Lemma 2.14,

$$[\![\Big(\mathsf{pkenc}^A \mathsf{pkdec}^{B_1} \cdots \mathsf{pkdec}^{B_n} \Big(\overset{q \text{ msgs}}{-\!\!\prec}_{\perp}, \leftarrow \bullet^n{}_{\perp} \Big) \mid \overset{q \text{ msgs}}{\Rightarrow\!\!\diamond\!\!?\!\!\Rightarrow\!\!\bullet\!\!\bullet}_{\perp} \Big)]\!] \leq [\![\mathbf{G}^{\text{WROB-CCA}}_{n,q}(\mathrm{PKE})]\!] \circ (\cdot \mathbf{C}).$$

Security. We now proceed to proving the security condition. Before we sketch the reductions from winning the underlying games, we remark that the simulation for corrupted receivers is always perfect (the ciphertext is computed exactly as in the honest encryption, and the injected ciphertexts are simply forwarded) and can hence be ignored in the following arguments. We use the notation

$$\mathbf{R}_q \coloneqq \mathsf{pkenc}^A \mathsf{pkdec}^{B_1} \cdots \mathsf{pkdec}^{B_n} \Big(\overset{q \text{ msgs}}{-\!\!\prec}, \leftarrow \bullet^n \Big) \quad \text{and} \quad \mathbf{S}_q \coloneqq \overset{q \text{ msgs}}{\Rightarrow\!\!\diamond\!\!?\!\!\Rightarrow\!\!\bullet\!\!\bullet}.$$

WROB-CCA. As a first intermediate step, we introduce a hybrid resource $\mathbf{H}_1$. This resource behaves like $\Rightarrow\!\diamond\!\Rrightarrow\bullet$, except that it allows for the delivery of an arbitrary message to a party other than the intended recipient: namely, instead of the input $E.l.j? \leftarrow \bar{k}$, $\mathbf{H}_1$ allows to deliver a message $\tilde{m}$ to a user B_j for $j \neq i_{\bar{k}}$ (still $m_{\bar{k}}$ for $j = i_{\bar{k}}$) by means of $E.l.j? \leftarrow (\bar{k}, \tilde{m})$. We use a modified simulator σ_1 that sends the decryption of the ciphertext simulated for message $\bar{k}$ under the key of user j. The systems $\sigma_1{}^E\mathbf{H}_1$ and $\mathbf{S}_q$ are equivalent unless, for some query, there is a user B_j, not the intended recipient of some ciphertext, that outputs a message upon receiving the ciphertext. If this situation occurs (i.e., some unintended recipient outputs a message from a ciphertext) it directly corresponds to winning the WROB-CCA game. In more detail, the reduction $\mathbf{C}'''$ obtains n generated keys at the inside interface (analogously to $\mathbf{C}$ above), which correspond to the users, and an additional key, used to simulate ciphertexts. It encrypts the plaintexts using the $\mathbf{enc}$-oracle of the game.

IND-CCA. We introduce a second hybrid $\mathbf{H}_2$ that behaves as $\mathbf{H}_1$ but additionally leaks the receiver's identity (no anonymity). The suitable simulator σ_2 always encrypts the all-zero string of appropriate length for the respective user, and decrypts as needed. Two things must be shown: first, that $\sigma_2{}^E\mathbf{H}_2$ is indistinguishable from $\mathbf{R}_q$; and second, that $\sigma_1{}^E\mathbf{H}_1$ and $\sigma_2{}^E\mathbf{H}_2$ are indistinguishable. We start with the former one. This is shown via a reduction $\mathbf{C}'$ that connects with the inside interface to the game $\mathbf{G}^{\text{PKE-CCA}}(\text{PKE})$. The reduction chooses a "target" receiver $\bar{k} \leftarrow\!\!\$\ [n]$ and a "target" message $\bar{\imath} \leftarrow\!\!\$\ [q]$ uniformly at random, and encrypts all messages to receivers with $k < \bar{k}$ as well as the messages $i < \bar{\imath}$ for receiver $\bar{k}$ as encryptions of the all-zero string (of the same length as the message), and all messages to receivers with $k > \bar{k}$ as well as the messages $i > \bar{\imath}$ for receiver $\bar{k}$ using the correct message and public key. The $\bar{\imath}$-th message to receiver $\bar{k}$ is encrypted using the challenge oracle, using the all-zero string as m_0 and the actual message as m_1. This can be seen as a hybrid argument as follows: Let $\mathbf{C}'_{\bar{k},\bar{\imath}}$ the reduction with fixed values $\bar{k} \in [n]$ and $\bar{\imath} \in [q]$, then

$$\mathbf{C}'_{\bar{k},\bar{\imath}}\mathbf{G}^{\text{PKE-CCA}}_{q,0}(\text{PKE}) \equiv \mathbf{C}'_{\bar{k},\bar{\imath}+1}\mathbf{G}^{\text{PKE-CCA}}_{q,1}(\text{PKE}) \quad \text{for} \quad \bar{\imath} < q \text{ and all } k \in [n],$$

and

$$\mathbf{C}'_{\bar{k},q}\mathbf{G}^{\text{PKE-CCA}}_{q,0}(\text{PKE}) \quad \equiv \quad \mathbf{C}'_{\bar{k}+1,1}\mathbf{G}^{\text{PKE-CCA}}_{q,1}(\text{PKE}) \qquad \text{for all} \qquad k \in [n-1].$$

Also,

$$\sigma_2{}^E\mathbf{H}_2 \equiv \mathbf{C}'_{n,q}\mathbf{G}^{\text{PKE-CCA}}_{q,0}(\text{PKE}) \quad \text{and} \quad \mathbf{R}_q \equiv \mathbf{C}'_{1,1}\mathbf{G}^{\text{PKE-CCA}}_{q,1}(\text{PKE}).$$

Consequently, by choosing the values $\bar{k} \in [n]$ and $\bar{\imath} \in [q]$ uniformly at random, the reduction loses a factor of nq.

IK-CCA. The last step is to show a reduction $\mathbf{C}''$ that turns a distinguisher between ${\sigma_1}^E\mathbf{H}_1$ and ${\sigma_2}^E\mathbf{H}_2$ corresponding, respectively, to the first and second hybrid introduced in the proof, into an IK-CCA-adversary. Recall that $\mathbf{H}_2$ behaves just like $\mathbf{H}_1$ except that it does not grant anonymity. We again use a hybrid argument with nq "intermediate" systems between ${\sigma_1}^E\mathbf{H}_1$ and ${\sigma_2}^E\mathbf{H}_2$, similarly to the IND-CCA case, such that each intermediate system embeds the challenge at a different position (as above). All other keys, encryptions, or decryptions are either simulated as "real" or as "ideal," depending on their position. The system where all the queries are "real" is equivalent to ${\sigma_2}^E\mathbf{H}_2$, and the system where only $\tilde{pk}$ was used (all queries are "ideal") is equivalent to ${\sigma_1}^E\mathbf{H}_1$. As in the case of IND-CCA, the reduction loses a factor of nq. □

4.7 Unilateral Key Establishment

Many practical security protocols used on the Internet are designed for a client-server setting where only the server has a certified public key. The most prominent example for this use case is access to web servers, but protocols for sending or receiving mail or for accessing database or directory servers often follow the same approach. In these settings, the client and the server establish a cryptographic key which has only *unilateral authentication*, i.e., the client is assured to share a key with the assumed server; the server has no comparable guarantee. The client is later authenticated by sending its credentials, often a username and a password, over a connection that is secured with the shared key.

In this section, we show that the unilateral key is *constructed* from an authenticated communication channel in one direction (say, from B to A) and an insecure communication channel in the opposite direction by a simple protocol based on a CPA-secure key-encapsulation mechanism (KEM). Indeed, the unilateral key formalizes the expected guarantee: In a setting where only one party, say B, can send messages authentically, the attacker can always block the messages sent by A and engage in the protocol with B, obtaining a key. We also show how the authenticated channel assumed by the protocol can be constructed in a setting where, e.g., a public-key infrastructure is available, and we show that the unilateral key is still a useful resource by proving the folklore belief that A can be authenticated later by, e.g., sending a password.

4.7.1 Constructing a Unilateral Key

A unilateral key $= \!=\!\bullet$ (as specified in System 24) can be constructed using a KEM scheme from a (single-message) insecure communication channel $- \rightarrow$ from A to B and an authenticated communication channel $\leftarrow \!-\!\bullet$ if the KEM scheme is CPA-secure. In more detail, we describe a protocol **kem** based on any KEM that achieves the construction

$$(- \rightarrow, \leftarrow \!-\!\bullet) \quad \xrightarrow{\textbf{kem}} \quad = \!=\!\bullet.$$

The protocol is a pair **kem** $= (\mathsf{decap}, \mathsf{encap})$ of converters and is based on a KEM scheme $\mathrm{KEM} = (\mathrm{KEMgen}, \mathrm{KEMenc}, \mathrm{KEMdec})$ as defined in Section 2.4.5. The protocol **kem** then behaves as follows.

decap: Initially, generate a key pair $(pk, sk) \leftarrow\!\!\$\ \mathrm{KEMgen}()$ and output pk at the inside sub-interface In.1.1?. Upon receiving the message (pk', c) at the inside sub-interface In.2.1!, check whether $pk' = pk$, and compute the key $k \leftarrow \mathrm{KEMdec}(sk, c)$. If the public key was correct, the computation succeeded, and the trigger at Out.1? has been given, output k at the outside interface Out.1!.

encap: Upon receiving a key pk' at the inside sub-interface In.1.1!, compute $(k, c) \leftarrow\!\!\$\ \mathrm{KEMenc}(pk')$ and output (c, pk') at the inside sub-interface In.2.1?. Once the trigger at Out.1? has been provided, output k at the outside interface Out.1!.

The client verifies the public key to detect whether it has been replaced by the attacker; this is necessary for the protocol to actually construct the key because the public key is transmitted over an insecure channel and may be changed by an attacker.

We prove the construction with respect to the simulator $\sigma_{\textbf{kem}}$ which behaves as follows:

1. Generate $(pk, sk) \leftarrow\!\!\$\ \mathrm{KEMgen}()$ and output pk at the first outside sub-interface.

2. Upon input at the first outside sub-interface:

   - On input $pk' = pk$, compute $(\bar{k}, c) \leftarrow\!\!\$\ \mathrm{KEMenc}(pk)$, output 0 at the inside B-sub-interface, and output (pk, c) at the second outside sub-interface.
   - On input $pk' \neq pk$, compute $(\bar{k}, c) \leftarrow\!\!\$\ \mathrm{KEMenc}(pk')$, output 1 at the inside B-sub-interface and $\bar{k}$ at the inside `key`-sub-interface, and output (pk', c) at the second outside sub-interface.

3. Upon receiving the trigger input $\uparrow$ at the second outside sub-interface, if pk was delivered in step 2, then output $\uparrow$ at the inside A-sub-interface.

Theorem 4.14. *For the protocol* $\mathsf{kem} = (\mathsf{decap}, \mathsf{encap})$ *described above and the simulator* σ_{kem}*,*

$$(- \rightarrow, \leftarrow \bullet) \quad \xrightarrow{\mathsf{kem}, \sigma_{\mathsf{kem}}, (0, \varepsilon)} \quad = \bullet,$$

with $\varepsilon \coloneqq [\![(\mathbf{G}_0^{\text{KEM-CPA}}(\text{KEM}) \mid \mathbf{G}_1^{\text{KEM-CPA}}(\text{KEM}))]\!] \circ (\cdot \mathbf{C})$ *and a reduction* $\mathbf{C}_{\mathsf{kem}}$ *described in the proof.*

Proof. For the availability condition, we show

$$\mathsf{decap}^A \mathsf{encap}^B \, (- \rightarrow_\perp, \leftarrow \bullet_\perp) \quad \equiv \quad = \bullet_\perp .$$

This follows directly from the correctness of the KEM scheme: for the key $k \in \mathcal{K}$ generated by $(k, c) \leftarrow \text{KEMenc}(pk)$, it holds that $k = \text{KEMdec}(sk, c)$.

For the security condition, we show that

$$\Big(\mathsf{decap}^A \mathsf{encap}^B \, (- \rightarrow, \leftarrow \bullet) \;\Big|\; \sigma_{\mathsf{kem}}{}^E = \bullet \Big) \\ = (\, \mathbf{C}_{\mathsf{kem}} \mathbf{G}_0^{\text{KEM-CPA}}(\text{KEM}) \mid \mathbf{C}_{\mathsf{kem}} \mathbf{G}_1^{\text{KEM-CPA}}(\text{KEM}) \,) ,$$

for a simple reduction $\mathbf{C}_{\mathsf{kem}}$ that is defined as follows: it obtains the public key pk at the inside interface and outputs it at the outside E-sub-interface. Upon input pk' at the outside E-sub-interface,

- if $pk' \neq pk$, then $\mathbf{C}_{\mathsf{kem}}$ computes $(k', c) \leftarrow\$ \text{KEMenc}(pk')$ and outputs c at the outside E-sub-interface and k' at the outside B-sub-interface.
- if $pk' = pk$, then $\mathbf{C}_{\mathsf{kem}}$ obtains (via $\mathbf{chgen}$) a pair (k, c) and outputs c at the outside E-sub-interface and k at the outside B-sub-interface.

In the case where $pk' \neq pk$ is delivered, the outputs in both cases are distributed equivalently. There is no output at the A-interface, the key $\bar{k}$ obtained by $\text{KEMenc}(pk')$ is output at the B-interface, and the value (pk', c) is output at the E-interface.

If pk is delivered, the distribution corresponds either exactly to the one given by $\mathbf{G}_0^{\text{KEM-CPA}}(\text{KEM})$ (when using the protocol), or to the one given by $\mathbf{G}_1^{\text{KEM-CPA}}(\text{KEM})$ (when using the resource $= \bullet$ and the simulator). Overall, this means that

$$\mathsf{decap}^A \mathsf{encap}^B \, (- \rightarrow, \leftarrow \bullet) \quad \equiv \quad \mathbf{C}_{\mathsf{kem}} \mathbf{G}_0^{\text{KEM-CPA}}(\text{KEM})$$

and

$$\sigma_{\mathsf{kem}}{}^{E}= =\!\bullet \quad \equiv \quad \mathbf{C}_{\mathsf{kem}}\mathbf{G}_1^{\text{KEM-CPA}}(\text{KEM})$$

which with Lemma 2.6 concludes the proof. □

Previous protocols for unilateral key establishment which are based on KEMs generally required stronger assumptions such as CCA security or random oracles. We emphasize, however, that the core of Shoup's A-DHKE protocol [Sho99] is a Diffie-Hellman key establishment, and if one interprets that as a KEM, then it is also used in this manner. A similar protocol based on a KEM with CPA-security is described in the preprint of Dodis and Fiore [DF13]; their (independent) work, however, has a different focus and uses game-based security definitions.

4.7.2 Constructing the Authenticated Channel

The construction in Section 4.7.1 assumes a pair of one single-use insecure and one single-use authenticated channel in opposite directions, in a setting with two parties (and one attacker). In a realistic scenario with multiple parties $A_1, \ldots, A_n$, however, such a (dedicated) authenticated channel from B to each A_i does not exist, and in this section we show how it is constructed from realistic resources.

Authenticated Transmission of Single Messages

The aim of a public-key infrastructure in the unilateral setting is to provide the parties $A_1, \ldots, A_n$ with an authentic copy of party B's public key. We model this functionality of the PKI as a resource that takes one message from B and distributes copies to $A_1, \ldots, A_n$, where the delivery of the copies may be delayed or prevented by the attacker at the E-interface. (Technically, the E-interface has for each $i \in [n]$ a sub-interface that accepts an $\uparrow$-message to provoke delivery.) The resource, which we denote as $\triangleright\!\!\diamond\!\!\bullet$, is specified with the same interface labels as the network, and is described in more detail as Systems 30 and 31. Of course, a protocol based on a PKI and certificates is only one possible way to construct this resource; any scheme that guarantees that B's public key is delivered authentically can be used.

The statement that a PKI constructs the resource $\triangleright\!\!\diamond\!\!\bullet$ with respect to the construction notion used here does not faithfully model the use of PKIs in current protocols. Intuitively, the reason is that the PKI is used by many different parties, and the modeling we use requires that all these parties behave according to the proven protocol, which is an unjustifiable assumption

System 30 The resource $\rhd\!\!-\!\diamond\!\!-\!\!\bullet_{\perp}$

upon $m \leftarrow B.1?$
  for $i \in [n]$ **do**
    $A_i.1! \leftarrow m$
  end for
end.

System 31 The resource $\rhd\!\!-\!\diamond\!\!-\!\!\bullet$ with accessible E-interface

upon $m \leftarrow B.1?$
  $E.1! \leftarrow m$
end.

once $(B.1? \neq \square) \wedge (E.i? = \uparrow)$
  $A_i.1! \leftarrow B.1?$
end.

for large-scale PKIs. One way of resolving these issues is by slightly adapting the construction notion along the lines of the recent work of Canetti et al. [CSV14]. A similar treatment following the paradigms used in our model would result in resources which are correlated (as random variables), we do, however, not pursue this question further here.

Authentication via Signatures

The resource $\rhd\!\!-\!\diamond\!\!-\!\!\bullet$ allows a party B to transmit one message authentically to all parties $A_1, \ldots, A_n$. To be useful in a key-establishment protocol where B has to send a different authenticated message to each A_i, however, we apply a signature scheme to "extend" this authentication to multiple transmissions. The resource $\Lleftarrow\!\!=\!\diamond\!=\!\!\bullet$ we construct has, as $\rhd\!\!-\!\diamond\!\!-\!\!\bullet$, one interface A_i for each $i \in [n]$, one interface B with sub-interfaces $B.i$ for each $i \in [n]$ (for sending a message to A_i), and one interface E, for the attacker. It is described in more detail as Systems 32 and 33.

System 32 The resource $\Lleftarrow\!\!=\!\diamond\!=\!\!\bullet_{\perp}$

upon $m \leftarrow B.i?$
  $A_i.1! \leftarrow m$
end.

The construction uses, in addition to the authenticated channel $\rhd\!\!-\!\diamond\!\!-\!\!\bullet$, a network of insecure channels which consists of, for each $i \in [n]$, one insecure

System 33 The resource $\Leftarrow\!\!\diamond\!\!-\!\!\bullet$ with accessible E-interface

$\mathcal{B} \leftarrow \emptyset$

upon $m \leftarrow B.i?$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{m\}$
  $E.i! \leftarrow m$
end.

once $(m \leftarrow E.i? \neq \square) \wedge (m \in \mathcal{B})$
  $A_i.1! \leftarrow m$
end.

single-message communication channel $\leftarrow -$ from B to A_i. This network is defined analogously to the resource ${}^{n}\!\leftarrow\!-\!\bullet$ in Section 4.6.1. The server's interface B has one sub-interface $B.i$ for each $i \in [n]$, which allows to send a message intended for A_i, and each interface A_i (potentially) outputs a message. The network can be described as the parallel composition of one resource $\leftarrow -$ per $i \in [n]$, where the A-interfaces of the channel (which has interfaces A, B, and E) is relabeled as A_i-interface and the B-interfaces are relabeled as $B.i$-sub-interface. We denote this resource as $\prod_{(I,J)\in\mathcal{P}} \leftarrow -$, with the set $\mathcal{P} := \{(A_i, B.i) : i \in [n]\}$ indicating the interfaces names. For brevity, we also use the notation ${}^{n}\!\leftarrow -$.

The protocol **sig** $= (\mathsf{vrf}, \ldots, \mathsf{vrf}, \mathsf{sig})$ with converters vrf (for $A_1, \ldots, A_n$) and sig (for B) use the signature scheme $\mathrm{SIG} = (\mathrm{SIGgen}, \mathrm{SIGsgn}, \mathrm{SIGvrf})$ in the following way:

sig**:** Initially, generate a key pair $(sk, vk) \leftarrow\!\!\$\ \mathrm{SIGgen}$ and output vk at the inside interface In.1.1? (i.e., to $\triangleright\!\!\diamond\!\!-\!\!\bullet$). Upon receiving a message $m \in \{0,1\}^*$ at the outside sub-interface Out.i?, compute $s \leftarrow\!\!\$\ \mathrm{SIGsgn}(m)$ and output (m, s) at the inside sub-interface In.2.i? (i.e., as directed to A_i).

vrf**:** Once the verification key vk is received at the inside sub-interface In.1.1! and a pair (m, s) is received at the inside sub-interface In.2.1!, compute $\mathrm{SIGvrf}\,(vk, m, s)$ and output m at the outside interface Out.1! if the verification succeeds.

For the security statement, we use the following simulator $\sigma_{\mathbf{sig}}$. Initially, $\sigma_{\mathbf{sig}}$ generates a key pair $(sk', vk') \leftarrow\!\!\$\ \mathrm{SIGgen}()$ and outputs vk' on the outside interface Out.1.1! (i.e., corresponding to $\triangleright\!\!\diamond\!\!-\!\!\bullet$). Upon a (trigger) input at the outside sub-interface Out.1.A_i?, $\sigma_{\mathbf{sig}}$ records A_i as "having received the

verification key." Whenever σ_{sig} obtains at the inside sub-interface $\mathsf{In}.i!$ for some $i \in [n]$ a message m', σ_{sig} computes $s' \leftarrow_{\$} \mathrm{SIGsgn}\left(sk', m'\right)$ and outputs (m', s') on the outside sub-interface $\mathsf{Out}.2.i!$ (corresponding to $\leftarrow -$ to A_i). Whenever σ_{sig} obtains at the outside sub-interface $\mathsf{Out}.2.i?$ for some $i \in [n]$ (corresponding to $\leftarrow -$ to A_i) the input (m', s'), if $\mathrm{SIGvrf}\left(vk', m', s'\right)$ succeeds and once there has been an input m at interface $\mathsf{In}.j!$ for some $j \in [n]$ and the key vk' has been delivered to A_i, then output m' at the inside sub-interface $\mathsf{In}.i?$.

Theorem 4.15. *Let* $\mathrm{SIG} = (\mathrm{SIGgen}, \mathrm{SIGsgn}, \mathrm{SIGvrf})$ *be a signature scheme and* $\mathbf{sig} = (\mathsf{vrf}, \mathsf{sig})$ *be the pair of converters described above. For the simulator* $\sigma_{\mathbf{sig}}$,

$$(\triangleright\!\!-\!\!\diamond\!\!-\!\!\bullet,\ {}^{n}\!\leftarrow -) \quad \xRightarrow{\mathbf{sig},\sigma_{\mathbf{sig}},(0,\varepsilon)} \quad \Leftarrow\!\!\!\!-\!\!\diamond\!\!-\!\!\bullet,$$

with $\varepsilon \coloneqq [\![\mathbf{G}_n^{\text{EUF-CMA}}(\mathrm{SIG})]\!] \circ (\cdot\mathbf{C}_{\mathbf{sig}})$ *and the reduction* $\mathbf{C}_{\mathbf{sig}}$ *described in the proof.*

Proof. For the availability condition, we show that

$$\left(\prod_{i\in[n]} \mathsf{vrf}^{A_i}\right) \mathsf{sig}^{B}\,(\triangleright\!\!-\!\!\diamond\!\!-\!\!\bullet_{\perp},\ {}^{n}\!\leftarrow -_{\perp}) \quad \equiv \quad \Leftarrow\!\!\!\!-\!\!\diamond\!\!-\!\!\bullet_{\perp},$$

which follows directly from the correctness of the signature scheme.

For the security condition, we use the notation

$$\mathbf{R} \quad \coloneqq \quad \left(\prod_{i\in[n]} \mathsf{vrf}^{A_i}\right) \mathsf{sig}^{B}\,(\triangleright\!\!-\!\!\diamond\!\!-\!\!\bullet,\ {}^{n}\!\leftarrow -) \qquad \text{and} \qquad \mathbf{S} \quad \coloneqq \quad \sigma_{\mathbf{sig}}{}^{E} \Leftarrow\!\!\!\!-\!\!\diamond\!\!-\!\!\bullet$$

and show that

$$[\![(\mathbf{R} \mid \mathbf{S})]\!] \quad \leq \quad [\![\mathbf{G}_n^{\text{EUF-CMA}}(\mathrm{SIG})]\!] \circ (\cdot\mathbf{C}_{\mathbf{sig}})$$

for the reduction $\mathbf{C}_{\mathbf{sig}}$ which, when connecting with its inside interface to $\mathbf{G}_n^{\text{EUF-CMA}}(\mathrm{SIG})$, simulates the same behavior as $\mathbf{R}$ at the outside interface. In more detail $\mathbf{C}_{\mathbf{sig}}$ first outputs the verification key vk obtained from the game at the left $E.1.1!$-sub-interface. Upon input a message m_i at some left $A.1.i?$, $\mathbf{C}_{\mathbf{mac}}$ queries $\mathbf{sign}(m_i)$ at the game, obtaining a signature s, and outputs (m, s) at the left $E.2.i!$-sub-interface. Upon input a pair (m', s') at the left $E.2.i?$-sub-interface, $\mathbf{C}_{\mathbf{sig}}$ verifies the signature and, in case of success, outputs m' at the left $A_i.1!$-sub-interface. The systems $\mathbf{R}$, $\mathbf{S}$, and $\mathbf{C}_{\mathbf{sig}}\mathbf{G}_n^{\text{EUF-CMA}}(\mathrm{SIG})$ are game-equivalent with respect to an MBO that formalizes that a forgery occurs, and using Lemma 2.14 concludes the proof. $\square$

Creating Independent Sessions

To use independent instances of the KEM-based protocol for multiple different clients, we need to separate the "sessions" of the protocol that correspond to different honest clients. This is done in a "session protocol" **sess** $= (\mathsf{sc}_1, \ldots, \mathsf{sc}_n, \mathsf{ss})$ where B, when sending a message to A_i, includes the "name" i when sending a message authentically via $\Leftarrow\!\!\diamond\!\!-\!\!\bullet$; this allows A_i to determine whether it is the intended receiver of an authenticated message.

We denote the resource constructed by this protocol as $\prod_{(I,J)\in\mathcal{P}} \leftarrow\!\!-\!\!\bullet$, the notation is to be understood as follows. For each pair $(I, J) \in \mathcal{P}$ there is one instance of the channel $\leftarrow\!\!-\!\!\bullet$ where the A-interface of the channel is the I-interface of the combined resource, and the B-interface of the channel is the J-interface of the combined resource. As the set is instantiated by pairs $(A_i, B.i)$, the combined resources has interfaces $A_1, \ldots, A_n$ and B, where the interface B has sub-interfaces $B.1, \ldots, B.n$.

The protocol **sess** that achieves the construction is specified as follows.

sc: Upon receiving a message m' at the inside interface (i.e., from $\Leftarrow\!\!\diamond\!\!-\!\!\bullet$) such that $m' = (i, m'')$, output m'' at the outside interface.

ss: Upon receiving an input m' at the outside i-sub-interface, send (i, m') via the inside i-sub-interface.

The security statement is proven with respect to the simulator σ_{sess} that, when it obtains a message at its inside interface, behaves as follows. For a message m' corresponding to the authenticated channel to A_i, output (i, m') as transmitted via $\Leftarrow\!\!\diamond\!\!-\!\!\bullet$ to A_i. On receiving messages at the outside interface, σ_{sess} behaves as follows: Upon delivery of a message (i, m) via $\Leftarrow\!\!\diamond\!\!-\!\!\bullet$ to A_i, output the trigger input at the inside sub-interface corresponding to the authenticated channel to A_i. (If the value i in the message does not match the index of A_i, drop the message and do not process further messages to A_i.)

Theorem 4.16. *For the protocol* **sess** $= (\mathsf{sc}_1, \ldots, \mathsf{sc}_n, \mathsf{ss})$ *described above and simulator* σ_{sess}*,*

$$\Leftarrow\!\!\diamond\!\!-\!\!\bullet \quad \xrightarrow{\mathsf{sess},\sigma_{\mathsf{sess}},(0,0)} \quad \prod_{(I,J)\in\mathcal{P}} \leftarrow\!\!-\!\!\bullet,$$

where the set $\mathcal{P}$ *is defined as* $\mathcal{P} \coloneqq \{(A_i, B.i) : i \in [n]\}$.

Proof. The validity of the availability condition, i.e., the statement

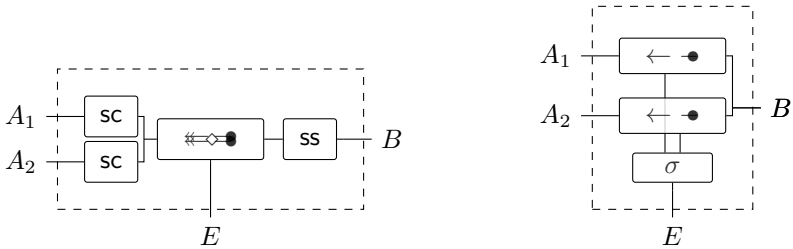
$$\left(\prod_{i\in[n]} \mathsf{sc}_i{}^{A_i}\right) \mathsf{ss}^B \Leftarrow\!\!\diamond\!\!-\!\!\bullet_{\perp} \quad \equiv \quad \prod_{(I,J)\in\mathcal{P}} \leftarrow\!\!-\!\!\bullet_{\perp}$$

follows from the correctness of the protocol. The converter ss, upon input a message m at the outside sub-interface i, sends the pair of message m and i, which the converter sc_i verifies. Since the verification always succeeds, messages are always output at the correct interface A_i

The security condition corresponds to the equivalence

$$\left(\prod_{i\in[n]} \mathsf{sc}_i{}^{A_i}\right) \mathsf{ss}^{B} \Leftarrow\!\!\diamond\!\!-\!\!\bullet \quad \equiv \quad \sigma_{\mathsf{sess}}{}^{E} \left(\prod_{(I,J)\in\mathcal{P}} \leftarrow\!\!-\!\!\bullet\right).$$

As each input to one system leads to exactly the same output as in the other system, the claimed equivalence holds. □



(a) Splitting the multi-channel into sessions.

(b) Parallel authenticated channels.

Figure 4.9: The session protocol splitting the multi-channel into a parallel composition of single channels.

Each of the constructed channels $\leftarrow\!\!-\!\!\bullet$ from $B.i$ to A_i can now be used together with an insecure channel from A_i to B_i for using one instance of the unilateral key establishment protocol. The composition theorems allow to deduce the corresponding security bounds for a protocol consisting of multiple steps or sessions.

Using names or nonces for separating sessions. The protocol **sess** uses the identifiers $i \in [n]$ of the parties $A_1, \ldots, A_n$ to "separate" the sessions by sending the messages together with the value i—a converter sc_i will accept a message only if it contains the corresponding value. This approach can be generalized by assuming a resource that outputs a unique value to each of the parties $A_1, \ldots, A_n$. In this case, the converter sc need not be parametrized and sends the obtained value to B initially. The resource providing the unique values can then be constructed probabilistically by having all parties $A_1, \ldots, A_n$ choose a random nonce.

4.7.3 Authenticating a Unilateral Key

If A and B share a password (which can be viewed as a key with low entropy), one can even construct a fully secure key from a unilateral one. (Note that a password cannot be directly used as a key in cryptographic schemes, because the contained entropy is too small.) The protocol is straightforward: A sends the password one-time pad encrypted using some part of the unilateral key, and outputs the unused part of the key as the key to be used by higher-level schemes. B decrypts (using the corresponding part of the unilateral key) and verifies the password; if the verification fails, all further messages are dropped. If the verification succeeds, B outputs the unused part of the key. Additionally, we let both parties output the password along with the key; this formalizes that the same password can be used to authenticate several keys (by the composition theorem).

Let Q be a distribution over $\{0,1\}^l$ with a certain maximum guessing probability $\varepsilon \in [0,1]$;[9] this models the assumption about the distribution of passwords. Then we denote by $\mathbf{Q}$ the resource that chooses a password $q \leftarrow \mathsf{Q}$ according to the distribution and outputs it at the A- and B-interfaces; the resource is defined analogously to $\bullet\!= =\!\bullet$ with the particular distribution Q. We then denote by $\mathbf{PWD} = (\mathsf{pwd\text{-}snd}, \mathsf{pwd\text{-}chk})$ the simple protocol sketched above, which in more detail works as follows.

pwd-snd: The converter initially outputs triggers at the inside sub-interfaces $\mathsf{In}.1.1?$ and $\mathsf{In}.2.1?$. Upon receiving $k \leftarrow \mathsf{In}.1.1!$ and $q \leftarrow \mathsf{In}.2.1!$, pwd-snd splits $k_1.k_2 \leftarrow k$ with $|k_1| = l$, computes $c \leftarrow q \oplus k_1$, and outputs c at the inside sub-interface $\mathsf{In}.3.1?$ and k_2 and q at the outside sub-interfaces $\mathsf{Out}.1!$ and $\mathsf{Out}.2!$, respectively.

pwd-chk: The converter initially outputs triggers at the inside sub-interfaces $\mathsf{In}.1.1?$ and $\mathsf{In}.2.1?$. Upon receiving $k \leftarrow \mathsf{In}.1.1!$, $q \leftarrow \mathsf{In}.2.1!$, and $c' \leftarrow \mathsf{In}.3.1!$, pwd-chk splits $k_1.k_2 \leftarrow k$ with $|k_1| = l$, compares $c' = q \oplus k_1$, and if the values are equal outputs k_2 and q at the outside sub-interfaces $\mathsf{Out}.1!$ and $\mathsf{Out}.2!$, respectively.

The construction is achieved relative to a simulator $\sigma_{\mathbf{PWD}}$ that behaves as follows. Obtaining an input $r \leftarrow \mathsf{Out}.1.B? \in \{0,1\}$ at the outside B-sub-interface corresponding to $= =\!\bullet$:

- if $r = 0$, then upon input trigger values at $\mathsf{Out}.1.A?$ and $\mathsf{Out}.2.A?$, simulate a uniformly random string $c \in \{0,1\}^l$ at the outside sub-

[9] We assume all passwords to be of the same length, the protocol extends to passwords with variable length by applying a suitable padding, which can be seen as an additional construction step.

interface Out.3.1! and output the trigger value at the inside sub-interfaces In.1.A? and In.2.A? (i.e., to $\mathbf{Q}$ and $\bullet\!\!=\ =\!\!\bullet$). Upon input a string $c' \in \{0,1\}^*$ at the outside sub-interface corresponding to $- \rightarrow$, if $c = c'$ then output the trigger value at the inside sub-interfaces In.1.B? and In.2.B? (i.e., of $\mathbf{Q}$ and $\bullet\!\!=\ =\!\!\bullet$).

- if $r = 1$, accept inputs but otherwise remain inactive.

The described protocol indeed achieves the described construction; the failure probability is exactly the attacker's probability of guessing the password. Note that in case the attacker shares the unilateral key with B, the password is not output and hence cannot be used in further protocol sessions.

Theorem 4.17. *Let* $\mathbf{Q}$ *be a (password) distribution with maximum guessing probability* ε*. Then for the simulator* σ_{PWD}*,*

$$(= =\!\!\bullet, \mathbf{Q}, - \rightarrow) \quad \xRightarrow{\mathsf{PWD},\sigma,\varepsilon} \quad (\bullet\!\!=\ =\!\!\bullet, \mathbf{Q}),$$

where the symbols correspond to families of resources $\overset{\kappa\text{-bit}}{=\ =\!\!\bullet}$*,* $\mathbf{Q}_l$*,* $\overset{l\text{ bits}}{- \rightarrow}$*, and* $\overset{\kappa-l\text{-bit}}{\bullet\!\!=\ =\!\!\bullet}$*.*

Proof. The availability condition, that is,

$$\mathsf{pwd\text{-}snd}^A \mathsf{pwd\text{-}chk}^B\,(= =\!\!\bullet_\perp, \mathbf{Q}_\perp, - \rightarrow_\perp) \quad \equiv \quad (\bullet\!\!=\ =\!\!\bullet_\perp, \mathbf{Q}_\perp)$$

follows since the same password and key are given to A and B, and by the correctness of the one-time pad analogously to Theorem 4.4.

For the security condition, we show that if $E.B? = 0$, then the systems $\mathsf{pwd\text{-}snd}^A \mathsf{pwd\text{-}chk}^B\,(= =\!\!\bullet, \mathbf{Q}, - \rightarrow)$ and $\sigma^E \bullet\!\!=\ =\!\!\bullet$ exhibit the same behavior. Once the trigger input for A is provided at the E-interface, a password and a uniformly random $(\kappa - l)$-bit key is output at the A-interface and a uniformly random l-bit string is output at the E-interface (at the sub-interface corresponding to $- \rightarrow$). If the same string is forwarded, the password and the $(\kappa - l)$-bit key are also output at the B-interface, otherwise nothing is output there. This follows because, within pwd-chk, a different ciphertext $c' \neq c$ will necessarily lead to a different result in the decryption, and hence the verification of the password will fail. The simulator achieves the same behavior by its definition.

To prove the remaining case, we define a monotone binary output (MBO) on the random system described by $\mathsf{pwd\text{-}snd}^A \mathsf{pwd\text{-}chk}^B\,(= =\!\!\bullet, \mathbf{Q}, - \rightarrow)$ that becomes 1 once $E.B? = 1$ and the attacker guesses the correct password, i.e., for the values $k' \in \{0,1\}^\kappa$ and $c' \in \{0,1\}^*$ provided to $= =\!\!\bullet$ and $- \rightarrow$, it holds that $|c'| = l$ and the XOR with the first l bits of k is equal to the password q. The system $\mathsf{pwd\text{-}snd}^A \mathsf{pwd\text{-}chk}^B\,(= =\!\!\bullet, \mathbf{Q}, - \rightarrow)$ with the described

MBO is conditionally equivalent to $\sigma^E \bullet\!\!= \!\!=\!\!\bullet$ as it is equivalent for $E.B? = 0$ and for $E.B? = 1$ given that the password guess is wrong. Hence, using Theorem 2.16, it follows that the distinguishing advantage is bounded by the probability ε of guessing the password. □

The constructed key can now of course be used as an assumed resource in any further construction, in particular one can now construct a secure channel by use of a MAC and a symmetric encryption scheme as discussed in Sections 4.3 and 4.4.

Practical protocols based on TLS, if used in a scenario with unilateral authentication, follow a different approach. The unilateral key obtained there is used to construct a pair of multiple-use secure channels which are unilateral but consistent in the sense that the server either communicates with the client, or the server communicates with the attacker. This pair of channels can then be authenticated (similarly to the example protocol shown here) by having the client send the password over the unilateral channel. The security proof then formalizes the intuitive argument that if the server receives the password, it can immediately conclude that it communicates with the client. The error term, as in the protocol above, includes the guessing probability for the assumed password distribution. This approach is sketched in more detail by Maurer et al. [MTC13].

Chapter 5

A Constructive Perspective on the TLS Record Layer

The TLS protocol is the most widely deployed cryptographic protocol on the Internet; its goal is to provide secure communication between two computers. The protocol can conceptually be decomposed into two phases: In the first phase, the *handshake protocol*, the two parties establish a secret key. In the second phase, the *record-layer protocol* uses the established key to protect the transmission of payload messages. In this section, we focus on proving the security of the record-layer protocol based on the assumption that a key has already been established.

The record-layer protocol defines multiple different cipher suites, the one that is actually used in a protocol session is negotiated during the handshake protocol. The most common cipher suites are based on the Authenticate-then-Encrypt transformation of a symmetric encryption scheme and a message authentication code, but modes based on authenticated encryption are also defined. While vulnerabilities [Rog95, Bar04] in the original specification and the general sensitivity of AtE to side-channel attacks [Vau02, AP12, AP13] have led to various practical attacks on the protocol, recent versions of the protocol can be proven secure (in models which do not capture timing attacks). In this section, we prove that the cipher suites based on Authenticate-then-Encrypt are indeed secure. The material in this section has been adapted from its original publication [MT10] to focus more on the analysis of the actual TLS protocol.

Following the constructive cryptography paradigm, the Authenticate-then-Encrypt modes of the TLS record-layer protocol can be seen as first constructing a confidential (but malleable) channel using the encryption scheme, and

then constructing a secure channel by use of a MAC. While Authenticate-then-Encrypt is not secure in general [Kra01, BN08], for the schemes employed in TLS the protocol is still secure [Kra01, MT10, PRS11].

5.1 Encryption

The TLS protocol as described in RFC 5246 [DR08] specifies two different types of encryption for the AtE modes: The stream cipher RC4, and block ciphers (3DES and AES) in CBC mode. We show for both modes that they construct a confidential channel which is still malleable, where the exact type of malleability depends on the encryption scheme.

5.1.1 Cipher Suites Based on Stream Ciphers

As discussed in Section 4.4.1, a stream cipher for a single message can be seen as a one-time pad encryption with a pseudo-random key stream. TLS uses a continuous key stream to encrypt a sequence of messages, where for each message a distinct subsequent part of the stream is used. Since the stream cipher is used on an insecure channel, the constructed confidential channel exhibits *XOR-malleability* as sketched in System 17, but further effects originating from variable message lengths and multiple messages must be considered in the case of TLS.

The stream cipher mode of TLS is described as a pair of converters $\mathbf{motp} = (\mathsf{motp\text{-}enc}, \mathsf{motp\text{-}dec})$ which, at the inside interface, connects to a key stream $\langle \bullet\!\overset{\text{1-bit}}{=\!=}\!\bullet \rangle$ and an insecure multiple-use channel $- \twoheadrightarrow$. Multiple messages are encrypted (and decrypted) using subsequent parts of the key stream. The converters are in more detail described as follows; they process the messages in order.

$\mathsf{motp\text{-}enc}$: Upon input the i-th message $m_i \leftarrow \mathsf{Out}.i?$, encrypt it by adding bit-by-bit the key bits with indices $r = \left(\sum_{j=1}^{i-1} |m_j|\right) + 1, \ldots, \left(\sum_{j=1}^{i} |m_j|\right) + 1$ obtained at the respective sub-interfaces $\mathsf{In}.1.r!$ (obtained upon input the respective triggers at $\mathsf{In}.1.r?$) and output the computed ciphertext at sub-interface $\mathsf{In}.2.i?$.

$\mathsf{motp\text{-}dec}$: Upon receiving the i-th ciphertext $c'_i \leftarrow \mathsf{In}.2.i!$, decrypt the message by adding bit-by-bit the key bits with indices $r = \left(\sum_{j=1}^{i-1} |c'_j|\right) + 1, \ldots, \left(\sum_{j=1}^{i} |c'_j|\right) + 1$ obtained at the respective sub-interfaces $\mathsf{In}.1.r!$ (upon input the respective triggers at $\mathsf{In}.1.r?$) and output the computed plaintext at the sub-interface $\mathsf{Out}.i!$.

As the ciphertexts are transmitted via an insecure channel, the lengths of the received ciphertexts may differ from the lengths of the messages that were encrypted. As subsequent messages are encrypted and decrypted using subsequent parts of the key stream, the malleability of the channel cannot be described as a transformation on individual messages.

The XOR-malleable multiple-use channel. In the XOR-malleable multiple-use channel $\multimap\!\!\oplus\!\twoheadrightarrow\!\bullet$, which is described as System 34, all inputs $A.i?$, $E.i?$ and outputs $B.i!$, $E.i!$ take values in $\{0,1\}^*$. It is not immediately clear that the channel $\multimap\!\!\oplus\!\twoheadrightarrow\!\bullet$ should be considered confidential, because the computation of $E.i!$ indeed involves bits of the plaintext message. Those are XORed, however, with a bit string that is uniformly random from the perspective of the attacker at the E-interface. In the following, for a string $x = x_1 \cdots x_n \in \{0,1\}^*$ and $i \leq n$, we refer by $x[i] = x_i$ to the i-th element of the string x. Furthermore, we use the notation $x[i:j] \coloneqq (x_i, \ldots, x_j)$, with the shorthands $x[:i] \coloneqq x[1:i]$ and $x[-i:] \coloneqq x[n-i+1:n]$, to refer to sub-strings of the string x.

The description of the channel is rather complicated and the only apparent application of the channel is to use it as an assumed resource in a protocol that is based on a MAC and constructs a secure channel. The complexity of the description appears to be inherent to the specified guarantee and due to the fact that the messages have variable length and are treated as a bit stream. Encryption schemes such as the one-time pad or stream ciphers should generally be used on an authenticated channel.

The construction of the XOR-malleable channel. The protocol **motp** constructs an XOR-malleable multiple-use channel from a key stream $\langle \bullet\!\overset{\text{1-bit}}{=\!=}\!\bullet \rangle$ and an insecure multiple-use channel $- \twoheadrightarrow$. The simulator $\sigma_{\textbf{motp}}$ used for this statement is simple: It basically forwards the messages between the inside- and outside interfaces. In more detail,

- Upon input $c_i \leftarrow \mathsf{In}.i! \in \{0,1\}^*$, once all trigger inputs $\mathsf{Out}.1.r.A?$ with r between $\sum_{k=1}^{j-1} |c_k| + 1$ and $\sum_{k=1}^{j} |c_k|$ have been provided, output $\mathsf{Out}.2.i! \leftarrow c_i$.
- Upon input $c'_i \leftarrow \mathsf{Out}.2.i?$, once all $\mathsf{Out}.2.j? \neq \square$ for $j < i$ and all trigger inputs $\mathsf{Out}.1.r.B?$ with r between $\sum_{k=1}^{j-1} |c'_k| + 1$ and $\sum_{k=1}^{j} |c'_k|$ have been provided, output $\mathsf{In}.i? \leftarrow c'_i$.

With respect to this simulator $\sigma_{\textbf{motp}}$, we prove that the multiple-use one-time pad constructs the channel $\multimap\!\!\oplus\!\twoheadrightarrow\!\bullet$ from a stream of key bits and a multiple-use insecure channel.

System 34 Continuously XOR-malleable multiple-use channel $\dashv\!\oplus\!\twoheadrightarrow\!\bullet$

$L_{\text{in}} \leftarrow 0$; $L_{\text{out}} \leftarrow 0$; $Q_{\text{bits}} \leftarrow \epsilon$

once $(\forall j \leq i : A.j? \neq \square) \wedge (\forall j < i : E.j! \neq \square)$ ▷ All previous messages are processed.
  $\mu \leftarrow \min\{\max\{L_{\text{out}} - L_{\text{in}}, 0\}, |A.i?|\}$ ▷ Number of "determined" bits.
  $\nu \leftarrow |A.i?| - \mu$ ▷ Number of "fresh" bits.
  if $\mu > 0$ **then** ▷ Bits determined by previous output.
    $s_1 \leftarrow (Q_{\text{bits}}[L_{\text{in}} + 1 : L_{\text{in}} + \mu] \oplus (A.i?)[1 : \mu])$
  end if
  if $\nu > 0$ **then** ▷ Fresh bits are chosen at random
    $s_2 \leftarrow_{\$} \{0,1\}^{\nu}$
    $Q_{\text{bits}} \leftarrow Q_{\text{bits}}.(s_2 \oplus A.i?[-\nu :])$
  end if
  $L_{\text{in}} \leftarrow L_{\text{in}} + |A.i?|$
  $E.i! \leftarrow s_1.s_2$
end.

once $(\forall j \leq i : E.j? \neq \square) \wedge (\forall j < i : B.j! \neq \square)$
  $\mu \leftarrow \min\{\max\{L_{\text{in}} - L_{\text{out}}, 0\}, |E.i?|\}$ ▷ Number of "determined" bits.
  $\nu \leftarrow |E.i?| - \mu$ ▷ Number of "fresh" bits.
  if $\mu > 0$ **then** ▷ Available bits are computed by masking.
    $s_1 \leftarrow (Q_{\text{bits}}[L_{\text{out}} + 1 : L_{\text{out}} + \mu]) \oplus (E.i?)[1 : \mu]$
  end if
  if $\nu > 0$ **then** ▷ Fresh bits are chosen at random.
    $s_2 \leftarrow_{\$} \{0,1\}^{\nu}$
    $Q_{\text{bits}} \leftarrow Q_{\text{bits}}.(s_2 \oplus E.i?[-\nu :])$
  end if
  $L_{\text{out}} \leftarrow L_{\text{out}} + |E.i?|$
  $B.i! \leftarrow s_1.s_2$
end.

Theorem 5.1. *The multiple-use one-time pad protocol* **motp** *constructs the continuously XOR-malleable channel* $-\!\!\oplus\!\twoheadrightarrow\!\bullet$ *from the insecure channel* $-\twoheadrightarrow$ *and a key stream* $\langle \bullet\!\!=\!\!=\!\!\bullet \rangle$*. For the simulator* $\sigma_{\mathbf{motp}}$*,*

$$\left(\left\langle \bullet\overset{\text{1-bit}}{=\!=}\bullet \right\rangle, -\twoheadrightarrow\right) \quad \xRightarrow{\mathbf{motp},\sigma_{\mathbf{motp}},(0,0)} \quad -\!\!\oplus\!\twoheadrightarrow\!\bullet,$$

where the construction is with respect to sequences $\langle \bullet\overset{\text{1-bit}}{=\!=}\bullet \rangle_{[\ell]}$*,* $-\overset{\ell\text{ bits}}{\twoheadrightarrow}$*, and* $-\!\!\oplus\!\overset{\ell\text{ bits}}{\twoheadrightarrow}\!\bullet$*, in* $\ell \in \mathbb{N}$*.*

Proof. For the correctness condition, the statement we want to prove is that for all $\ell \in \mathbb{N}$,

$$\mathsf{motp\text{-}enc}^A \mathsf{motp\text{-}dec}^B \left(\left\langle \bullet\overset{\text{1-bit}}{=\!=}\bullet_{\perp} \right\rangle_{[\ell]}, -\overset{\ell\text{ bits}}{\twoheadrightarrow}_{\perp}\right) \quad \equiv \quad -\!\!\oplus\!\overset{\ell\text{ bits}}{\twoheadrightarrow}\!\bullet_{\perp}.$$

Denote the key bit output by the i-th copy of the key resource as k_i. Then the messages $m_1, m_2, \ldots$ lead to the ciphertexts

$$m_1 \oplus (k_1 \ldots k_{|m_1|}), m_2 \oplus (k_{|m_1|+1} \ldots k_{|m_1|+|m_2|}), \ldots$$

generated by $\mathsf{motp\text{-}enc}$, while $\mathsf{motp\text{-}dec}$ adds the same key bits and outputs $m_1, m_2, \ldots$. The constructed resource simply forwards the messages. Both systems provide output until the accumulated length of inputs is longer than ℓ.

For the security condition, for all $\ell \in \mathbb{N}$, we show that

$$\mathsf{motp\text{-}enc}^A \mathsf{motp\text{-}dec}^B \left(\left\langle \bullet\overset{\text{1-bit}}{=\!=}\bullet \right\rangle_{[\ell]}, -\overset{\ell\text{ bits}}{\twoheadrightarrow}\right) \quad \equiv \quad \sigma_{\mathbf{motp}}{}^E -\!\!\oplus\!\overset{\ell\text{ bits}}{\twoheadrightarrow}\!\bullet.$$

We use the notation $\mathbf{R}$ for the left-hand side and $\mathbf{S}$ for the right-hand side of the equivalence.

For the resource $\mathbf{R}$, on inputs $m_1, m_2, \ldots$ at the A-interface, the output at the E-interface is $m_1 \oplus (k_1 \ldots . k_{|m_1|}), m_2 \oplus (k_{|m_1|+1} \ldots . k_{|m_1|+|m_2|}), \ldots$. The output at the B-interface on inputs $c'_1, c'_2, \ldots$ at the E-interface is $c'_1 \oplus (k_1 \ldots . k_{|c'_1|}), c'_2 \oplus (k_{|c'_1|+1} \ldots . k_{|c'_1|+|c'_2|}), \ldots$, where all key bits are independent and distributed uniformly, which follows directly by the definition of **motp**.

To describe $\mathbf{S}$, consider a state where q_A messages $m_1, \ldots, m_{q_A}$ have been input at the A-interface, and q_E messages $c'_1, \ldots, c'_{q_E}$ have been input at the E-interface. We define the terms $L^A_j \coloneqq \sum_{\ell=1}^{j} |m_j|$ and $L^E_j \coloneqq \sum_{\ell=1}^{j} |c'_j|$. The concatenation of the messages at the A-interface is a bit string $m = m_1 \ldots . m_{q_A}$; so are the outputs $c = c_1 \ldots . c_{q_A}$ at the E-interface, the inputs $c' = c'_1 \ldots . c'_{q_E}$ at the E-interface, and the outputs $m' = m'_1 \ldots . m'_{q_E}$ at the B-interface. The output of $\mathbf{S}$ is distributed as follows:

- On input m_{q_A+1} at the A-interface,
  - For $L^A_{q_A} < i \leq \min\left\{L^A_{q_A+1}, L^E_{q_E}\right\}$, the output bit $c[i]$ is determined as $c[i] = m[i] \oplus c'[i] \oplus m'[i]$. The same equality holds in the case of $\mathbf{R}$.
  - For $L^E_{q_E} < i \leq L^A_{q_A+1}$, the bit $c[i]$ is uniformly random. This is the same in the case of $\mathbf{R}$, where the key bit k_i (and hence the respective ciphertext bit) is uniformly random.
- On input m_{q_E+1} at the E-interface,
  - For $L^E_{q_E} < i \leq \min\left\{L^E_{q_E}, L^A_{q_A}\right\}$, the output bit $m'[i]$ is determined as $m'[i] = m[i] \oplus c'[i] \oplus c[i]$. The same equality holds in the case of $\mathbf{R}$.
  - For $L^A_{q_A} < i \leq L^E_{q_E+1}$, the bit $m'[i]$ is uniformly random. This is the same in the case of $\mathbf{R}$, where the key bit k_i (and hence the respective message bit) is uniformly random.

The proof concludes by observing that also here, both systems provide output until the accumulated input length exceeds ℓ bits. As the systems are equivalent, the output of each distinguisher has the same distribution in both cases and hence the distinguishing advantage is 0. □

The XOR-malleable channel $-\!\!\oplus\!\!\twoheadrightarrow\!\!\bullet$ is not a useful resource on its own, but a (strongly unforgeable) MAC constructs from $-\!\!\oplus\!\!\twoheadrightarrow\!\!\bullet$ and a shared key $\bullet\!\!=\;=\!\!\bullet$ a fully secure channel, as shown in Section 5.2.

5.1.2 Cipher Suites Based on CBC-Mode Encryption

The CBC-mode encryption based on a shared URP has been described for a single message in Section 4.4.2. The extension to the multiple-message case is straightforward: For each message, one chooses a fresh initialization vector uniformly at random and then applies the same encryption mechanism as before. In early versions of SSL/TLS, the initialization vector was not chosen as a fresh block, but the final block of the previous ciphertext was used. This mistake was pointed out by Rogaway [Rog95] and Bard [Bar04] and lead to a vulnerability exploited in the BEAST attack [DR11]. The vulnerability was fixed in TLS 1.1.

We describe the multiple-use version of CBC-mode encryption based on an n-bit permutation as a pair of converters $\mathbf{mcbc}_n = (\mathsf{mcbc\text{-}enc}_n, \mathsf{mcbc\text{-}enc}_n)$ using the following notation for the inputs and outputs. Consistently with the notation in Section 4.4.2, we use $m_i = m_{i,1}.m_{i,2}\ldots$ to denote the plaintext

messages input to the encryption converter $\mathsf{mcbc\text{-}enc}_n$, which consist of n-bit blocks $m_{i,j} \in \{0,1\}^n$. We use $c_i = c_{i,0}.c_{i,1}.\ldots$ for the ciphertexts generated by $\mathsf{mcbc\text{-}enc}_n$ and $c'_i = c'_{i,0}.c'_{i,1}.\ldots$ for ciphertexts received by $\mathsf{mcbc\text{-}dec}_n$, and $m'_i = m'_{i,0}.m'_{i,1}.\ldots$ for plaintexts output by $\mathsf{mcbc\text{-}dec}_n$.

The block-malleable multiple-use channel. The malleability of the channel constructed by the protocol $\mathbf{mcbc}_n$ from a shared URP and an insecure channel resembles the block-oriented structure of the encryption scheme; it is formally described as System 35. The inputs and outputs at the E-interface have labels in $i \in \mathbb{N}$, and all inputs and outputs are bit strings of length $\ell \in n \cdot \mathbb{N}$.

As during the decryption in $\mathsf{mcbc\text{-}dec}_n$ the ciphertext is split into blocks, and each of the blocks (except for the first one) corresponds to one block of plaintext, each input at the E-interface of $-\!\!\boxed{n}\!\!\twoheadrightarrow\!\!\bullet$ can be viewed as describing an output at the B-interface in a block-wise way. Each output block $m'_{i,j} \in \{0,1\}^n$ at the B-interface is either (nearly) uniformly random (if the ciphertext block was not used before) or specified by a block $m_{r,s}$ or $m'_{r,s}$ from a previous message and a mask $x \in \{0,1\}^n$ (obtained as the XOR of two ciphertext blocks; this is due to the fact that the previous ciphertext block is XORed to the subsequent plaintext block). Hence, the attacker can "apply masks" $x \in \{0,1\}^n$ such that $x = c_{r,s-1} \oplus c'_{i,j-1}$ (or $x = c'_{r,s-1} \oplus c'_{i,j-1}$), where $c'_{i,j-1}$ is the ciphertext corresponding to the preceding block (and may hence essentially be chosen if $j = 1$ or the preceding block does not correspond to a previously used one).

As for the XOR-malleable channel, the description of the channel is complicated and the only apparent application of the channel is to use it as an assumed resource in a protocol that is based on a MAC and constructs a secure channel.

Construction of the block-malleable channel. The protocol $\mathbf{mcbc}_n$ constructs the block-malleable multiple-use channel $-\!\!\boxed{n}\!\!\twoheadrightarrow\!\!\bullet$ from a shared URP $\overleftrightarrow{\mathbf{P}}_n$ and an insecure multi-message channel $- \twoheadrightarrow$. The construction is proven relative to a simulator $\sigma_{\mathbf{mcbc}}$ that (basically) forwards the messages between the inside interface and the second outside interface, with the exception that messages are forwarded from inside to outside once the input $\mathsf{Out}.1.A?$ has been provided, and messages are forwarded from outside to inside only once the input $\mathsf{Out}.1.B?$ has been provided. The statement is formalized in the following theorem.

Theorem 5.2. *The protocol* $\mathbf{mcbc}_n = (\mathsf{mcbc\text{-}enc}_n, \mathsf{mcbc\text{-}dec}_n)$ *constructs the* n*-bit block-malleable channel* $-\!\!\boxed{n}\!\!\twoheadrightarrow\!\!\bullet$ *from the insecure channel* $- \twoheadrightarrow$ *and shared*

System 35 Continuously block-malleable multiple-use channel $\dashv\!\boxed{n}\!\twoheadrightarrow\!\bullet$

$\mathcal{B}, \mathcal{B}' \leftarrow \emptyset$
$c_{*,-1}, c'_{*,-1} \leftarrow 0 \cdots 0 \in \{0,1\}^n$

once $(\forall j \leq i : A.j? \neq \square) \wedge (\forall j < i : E.j! \neq \square)$ ▷ First unprocessed one.
  $l \leftarrow \frac{|A.i?|}{n}$
  $m_{i,1}. \ldots .m_{i,l} \leftarrow A.i?$ ▷ Split the message into blocks.
  $c_{i,0} \leftarrow\!\$ \{0,1\}^n$ ▷ Choose the IV uniformly at random.
  for $j = 1, \ldots, l$ **do**
    $c_{i,j} \leftarrow\!\$ \{0,1\}^n \setminus (\mathcal{B} \cup \mathcal{B}')$ ▷ Sample blocks without collisions.
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{c_{i,j}\}$
  end for
  $E.i! \leftarrow c_{i,0}. \ldots .c_{i,l}$ ▷ Assemble the ciphertext.
end.

once $(\forall j \leq i : E.j? \neq \square) \wedge (\forall j < i : B.j! \neq \square)$
  $l \leftarrow \frac{|E.i?|}{n} - 1$ ▷ Plaintext message is one block shorter.
  $c'_{i,0}. \ldots .c'_{i,l} \leftarrow E.i?$ ▷ Split into n-bit strings.
  $m'_i \leftarrow \epsilon$
  for $j = 1, \ldots, l$ **do**
    if $c'_{i,j} \in \mathcal{B}$ **then** ▷ Block used in a message (not IV) before.
      Let $\bar{i}, \bar{j} \in \mathbb{N}$ such that $c_{\bar{i},\bar{j}} = c'_{i,j}$
      $m'_{i,j} \leftarrow m_{\bar{i},\bar{j}} \oplus c_{\bar{i},\bar{j}-1} \oplus c'_{i,j-1}$ ▷ Compute from previous
    else if $c'_{i,j} \in \mathcal{B}'$ **then** ▷ Block injected as message (not IV) before.
      Let $\bar{i}, \bar{j} \in \mathbb{N}$ such that $c'_{\bar{i},\bar{j}} = c'_{i,j}$ and $\bar{i} < i$
      $m'_{i,j} \leftarrow m'_{\bar{i},\bar{j}} \oplus c'_{\bar{i},\bar{j}-1} \oplus c'_{i,j-1}$ ▷ Compute from previous.
    else ▷ Fresh block (or used as IV).
      $m'_{i,j} \leftarrow\!\$ \{0,1\}^n$ ▷ Determine new value for URP.
      $\mathcal{B}' \leftarrow \mathcal{B}' \cup \{c'_{i,j}\}$
    end if
  end for
  $B.i! \leftarrow m'_{i,1}. \ldots .m'_{i,l}$ ▷ Assemble the message.
end.

uniform random permutation $\overleftrightarrow{\mathbf{P}}_n$*, with error* $(q\ell/n)^2/2^n$ *for* q *messages of length at most* ℓ*. For the simulator* σ_{mcbc}*,*

$$\left(\overleftrightarrow{\mathbf{P}}_n, - \twoheadrightarrow\right) \quad \xRightarrow{\mathsf{mcbc}_n, \sigma_{\mathsf{mcbc}}, (0, \varepsilon_{q,\ell,n})} \quad -\boxed{n}\twoheadrightarrow\bullet,$$

where the symbols refer to families $\overleftrightarrow{\mathbf{P}}_n^{|q\ell/n}$*,* $\overset{q\times(\ell+n)\text{ bits}}{- \twoheadrightarrow}$*, and* $\overset{q\times\ell\text{ bits}}{-\boxed{n}\twoheadrightarrow\bullet}$ *in* $q \in \mathbb{N}$ *and* $\ell \in n \cdot \mathbb{N}$ *with* $\varepsilon_{q,\ell,n} = 5(q\ell/n)^2/2^{n+1}$*.*

The channel $-\boxed{n}\twoheadrightarrow\bullet$ is not a monotone system according to Section 3.4.1. This is due to the fact that the blocks output at the E-interface are sampled uniformly, while the blocks output at the B-interface are computed to be consistent with the previous outputs at the E-interface. This can, however, be captured by a suitable MBO, such that the resulting system is monotone unless the MBO becomes 1. (This is not relevant in this proof but in the proof of Theorem 5.4.)

Proof. For the correctness condition, the statement to be proven is that for all $q, \ell \in \mathbb{N}$,

$$\mathsf{mcbc\text{-}enc}_n{}^A\mathsf{mcbc\text{-}dec}_n{}^B\left(\overleftrightarrow{\mathbf{P}}_n^{|q\ell/n}{}_{\perp}, \overset{q\times(\ell+n)\text{ bits}}{- \twoheadrightarrow}{}_{\perp}\right) \quad \equiv \quad \overset{q\times\ell\text{ bits}}{-\boxed{n}\twoheadrightarrow\bullet}{}_{\perp}.$$

This follows analogously to the correctness condition in Theorem 4.6.

For the security condition (again for all $q, \ell \in \mathbb{N}$), i.e.

$$\left(\mathsf{mcbc\text{-}enc}_n{}^A\mathsf{mcbc\text{-}dec}_n{}^B\left(\overleftrightarrow{\mathbf{P}}_n^{|q\ell/n}, \overset{q\times(\ell+n)\text{ bits}}{- \twoheadrightarrow}\right) \;\middle|\; \sigma_{\mathsf{mcbc}}{}^E\overset{q\times\ell\text{ bits}}{-\boxed{n}\twoheadrightarrow\bullet}\right) \quad \leq \quad \varepsilon_{q,\ell,n},$$

we use the notation $\mathbf{R}$ for the left-hand side and $\mathbf{S}$ for the right-hand side of the distinction problem.

We extend both systems $\mathbf{R}$ and $\mathbf{S}$ by a monotone binary output (MBO). For $\mathbf{R}$, the MBO becomes 1 as soon as for an output $E.i!$, one of the blocks $c_{i,j}$ for $j \in [l_i]$ with $l_i \coloneqq \frac{|m_i|}{n}$ collides either with a previous block $c_{i',j'}$ with $i' \leq i$, $j' \in [l_{i'}]$ (with $l_{i'}$ defined analogous to l_i), or with a block $c'_{i',j'}$ of a message previously input as $E.i'?$. For $\mathbf{S}$, the MBO becomes 1 as soon as for two pairs of indices $(i,j) \neq (i',j')$, the values $c_{i,j-1} \oplus m_{i,j}$ (or $c'_{i,j-1} \oplus m'_{i,j}$) and $c_{i',j'-1} \oplus m_{i',j'}$ (or $c'_{i',j'-1} \oplus m'_{i',j'}$) coincide.

We define an additional system $\mathbf{H}$ that behaves essentially as $\mathbf{S}$ but samples the message blocks such that no collision of the type described for $\mathbf{S}$ occurs. Then, on the one hand, the conditional equivalence $\hat{\mathbf{S}} \models \mathbf{H}$ holds trivially by the definition of $\mathbf{H}$. On the other hand, $\hat{\mathbf{R}} \models \mathbf{H}$ holds as well. The distribution of outputs on an input at the E-interface is exactly the same, and

the distribution of the outputs of both systems on an input at the A-interface is, analogous to above, the uniform distribution on all admissible values.

To complete the proof, it is sufficient to analyze the probabilities of (non-adaptively) provoking the MBOs in the two systems.

Provoking the MBO in $\hat{\mathbf{R}}$: The MBO in $\hat{\mathbf{R}}$ formalizes collisions in the ciphertext blocks output at interface E. Such a collision occurs if and only if two inputs to the shared URP $\overleftrightarrow{\mathbf{P}}$ are equal, as $\mathbf{P}$ implements a permutation. The proof now proceeds as follows: The shared URP $\overleftrightarrow{\mathbf{P}}_n$ is replaced by a system $\overleftrightarrow{\mathbf{P}}'_n$ which builds a function table $P : \{0,1\}^n \to \{0,1\}^n$ and its inverse P^{-1} "on demand," answering consistently with previous queries if defined, and otherwise sampling each response to a query (independent of the direction) uniformly at random.

As a random system, $\mathbf{P}'_n$ keeps two (initially empty) partial mappings $P, P^{-1} \subset (\{0,1\}^n \to \{0,1\}^n)$. Upon an input $X_i = (\mathsf{fwd}, x)$ with $x \in \{0,1\}^n$, if $x \notin \operatorname{im} P$, then it samples $y \leftarrow\$ \{0,1\}^n$ and sets $P \leftarrow P \cup \{(x,y)\}$. If $P^{-1}(y) = \square$, then $P^{-1} \leftarrow P^{-1} \cup \{(y,x)\}$. Finally, it outputs $Y_i = P(x)$. Upon an input $X_i = (\mathsf{bwd}, y)$, if $y \notin \operatorname{im} P^{-1}$, then it samples $x \leftarrow\$ \{0,1\}^n$ and sets $P^{-1} \leftarrow P^{-1} \cup \{(y,x)\}$. If $P(x) = \square$, then set $P \leftarrow P \cup \{(x,y)\}$. Finally, it outputs $Y_i = P^{-1}(y)$. (Intuitively, contradictions between queries are resolved by giving precedence to the already defined value.)

We define

$$\mathbf{R}' \quad = \quad \mathsf{mcbc\text{-}enc}_n{}^A \mathsf{mcbc\text{-}dec}_n{}^B \left(\overleftrightarrow{\mathbf{P}}'_n{}^{|q\ell/n}, \overset{q \times (\ell+n)\text{ bits}}{- \twoheadrightarrow} \right)$$

as well as a monotone binary output $A = (A_1, A_2, \dots)$ on $\mathbf{P}'_n$ as follows. The MBO becomes 1 if

- there are i, j with $X_i = (\mathsf{fwd}, x) \neq X_j = (\mathsf{fwd}, x')$ but $Y_i = Y_j$,
- or there are i, j with $X_i = (\mathsf{bwd}, y) \neq (\mathsf{bwd}, y')$ but $Y_i = Y_j$;
- there are i, j with $X_i = (\mathsf{fwd}, x)$ and $X_j = (\mathsf{bwd}, y)$, such that $Y_i = y$ but $x \neq Y_j$,
- or there are i, j with $X_i = (\mathsf{fwd}, x)$ and $X_j = (\mathsf{bwd}, y)$, such that $Y_i \neq y$ but $x = Y_j$.

Then we obtain that $\mathbf{P}'_n{}^A \models \mathbf{P}_n$, since the only way for $\mathbf{P}'_n$ to become inconsistent with $\mathbf{P}_n$ is by a collision that is exactly captured by the MBO A; the conditional equivalence holds as the MBO is defined only on the outputs. (As $\overleftrightarrow{\mathbf{P}}_n$ and $\overleftrightarrow{\mathbf{P}}'_n$ differ from $\mathbf{P}_n$ and $\mathbf{P}_n$ only by how the inputs and outputs

are assigned to interfaces, this also means that $\overleftrightarrow{\mathbf{P}}'_n{}^{A'} \models \overleftrightarrow{\mathbf{P}}_n$ with respect to an MBO A' which is a slightly modified definition of the MBO A according to the interfaces of $\overleftrightarrow{\mathbf{P}}'$.) As the other systems in $\mathbf{R}$ and $\mathbf{R}'$ do not affect the MBO, the conditional equivalence $\hat{\mathbf{R}}' \models \mathbf{R}$ with the same MBO defined on the sub-system $\overleftrightarrow{\mathbf{P}}'_n$ holds holds.

We define an additional MBO on $\mathbf{P}'_n$ (and $\mathbf{P}_n$) which formalizes a collision on the inputs. More formally, the MBO $B = (B_1, B_2, \dots)$ becomes 1 if there are i, j with $X_i = (\mathsf{fwd}, x) = X_j$ for some $x \in \{0,1\}^n$. Then $\mathbf{R}^B \models \mathbf{H}$, because the outputs at the E-interface will output blocks which are uniformly random, but without collisions. We can directly conclude that $\mathbf{R}'^{A' \vee B} \models \mathbf{H}$; both systems can be described as sampling the outputs based on lazy sampling of the URP.

Now we bound the performance $[\![\mathbf{R}^B]\!]$, where we clearly have

$$[\![\mathbf{R}^B]\!] \quad \leq \quad \left[\!\!\left[\mathbf{R}'^{A' \vee B}\right]\!\!\right],$$

as $[\![\mathbf{R}^B]\!]$ corresponds to the probability where in the system $\mathbf{R}'^{A' \vee B}$ the MBO B is provoked before the MBO A'. Hence, it is sufficient to bound the advantage $\left[\!\!\left[\mathbf{R}'^{A' \vee B}\right]\!\!\right]$.

As the probability of provoking the MBO $A' \vee B$ is bounded by the probability to sample an output that contradicts to the previously sampled part of a distribution and the undefined values in $\mathbf{P}'_n$ are sampled uniformly at random, we obtain the same bound as for the standard collision probability, hence

$$[\![(\mathbf{R} \mid \mathbf{H})]\!] \quad \leq \quad \left[\!\!\left[\mathbf{R}'^{A' \vee B}\right]\!\!\right] \quad \leq \quad \frac{(q\ell/n)^2}{2^{n-1}},$$

since overall up to $2q\ell/n$ queries are made to $\mathbf{P}'_n$.

Provoking the MBO in $\hat{\mathbf{S}}$: We define an MBO C on $\mathbf{S}$ that becomes 1 if any two blocks output at the E-interface collide. As in $\mathbf{H}$ the blocks are chosen uniformly at random such that no collision occurs and the systems otherwise behave equivalently, we obtain $\hat{\mathbf{S}} \models \mathbf{H}$. By the standard collision probability, the probability of provoking this MBO (and hence the distinguishing advantage) is bounded by $(q\ell/n)^2/2^{n+1}$.

Finally

$$[\![(\mathbf{R} \mid \mathbf{S})]\!] \quad \leq \quad [\![(\mathbf{R} \mid \mathbf{H})]\!] + [\![(\mathbf{H} \mid \mathbf{S})]\!] \quad \leq \quad \frac{5\,(q\ell/n)^2}{2^{n+1}},$$

which concludes the proof. □

The bound stated in the original publication [MT10] is different from the one obtained in the proof; the bound given in the statement is too tight by a factor of 2. The bound is not optimal: Paterson et al. [PRS11] describe bounds (for the composed scheme) in which the terms corresponding to the CBC-mode encryption are roughly comparable to the ones we describe here, but they achieve slightly better (constant) factors by a more detailed analysis. This can be viewed as that the MBO we describe in the proof of Theorem 5.2 is slightly too permissive.

5.2 Padding and Authentication

The second step in the SSL/TLS record-layer protocol takes care of the message authentication and is based on a MAC, sequence numbers, and a specific encoding. This sub-protocol can be seen as a construction of a secure channel from the malleable confidential channel and a secret key. A sequence number is included in the computation of the MAC to ensure that the messages arrive in the correct order. The plaintext and the MAC are then encoded in a specific format (which may include compression and also padding in the case of CBC-mode encryption) before the obtained string is transmitted over the confidential channel; this is why Paterson et al. [PRS11] refer to the mode as MEE (MAC-Encode-Encrypt).

We specify these protocol steps of the protocol as a pair of converters **tls-mac** $= (\mathsf{tls\text{-}tag}, \mathsf{tls\text{-}chk})$, which is conceptually depicted in Figure 5.1. The protocol is based on the following components.

- A MAC function $\mathrm{mac} : \{0,1\}^k \times \{0,1\}^{\leq 17510} \to \{0,1\}^{160}$. In TLS 1.2, the MAC functions are based on HMAC with SHA-1, SHA-256, or (deprecated) MD5 [DR08].

- A sequence number in $\{0, \ldots, 2^{64} - 1\}$. While the MAC is computed on the plaintext message and the sequence number, the sequence number is *not* sent with the message. Rather, the receiver also computes the sequence number and uses the locally computed number in the verification of the MAC.

- A padding function $\mathrm{Enc} : \{0,1\}^{\leq 17510+160} \to \{0,1\}^{\leq 18432}$; the function must be injective to be decodable. We assume that Enc^{-1} returns $\diamond$ on values not in the range of Enc.

  The padding scheme used in TLS ensures that the length of authenticated plaintexts is a multiple of n bits. Given a plaintext message m

with $l \coloneqq |m|$, the length of the padded message generated by the basic version[1] of the TLS padding scheme is $\hat{l} = n\lceil (l+8)/l \rceil$ and the last $l^{\mathsf{byte}} = (\hat{l} - l)/8$ bytes are filled with the value l^{byte} (in standard encoding as an octet). The described variant is a function, and invalidly padded messages are rejected.

The MAC is computed on the string $\mathrm{macEnc}(m, i) = \langle i \rangle_{64} . x . \langle |m| \rangle_{16} . m$, where $\langle i \rangle_\ell$ means that $i \in \mathbb{N}$ is encoded as an ℓ-bit string, and x is a value that encodes the type of the message (i.e., handshake, payload, alert,...) and the version of the protocol in use.

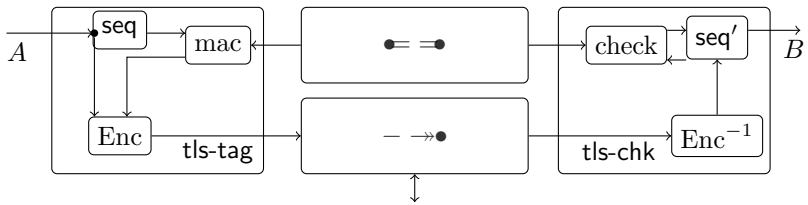


Figure 5.1: The authentication protocol **tls-mac** $=$ (tls-tag, tls-chk) is based on the MAC scheme $\mathrm{MAC} = (\mathrm{mac}, \mathrm{check})$. The systems seq and seq′ handle sequence numbers; Enc and Enc^{-1} convert the message-tag pairs for the message space of $\multimap\!\!\oplus\!\twoheadrightarrow\!\bullet$ and $\multimap\!\!\boxed{n}\!\twoheadrightarrow\!\bullet$.

5.2.1 Construction Based on the XOR-Malleable Channel

The protocol **tls-mac** guarantees that if the message output at the B-interface of the confidential channel is different from all messages input at the A-interface before, then it will either be rejected or results in the breach of the MAC security. The proof of the construction which is based on the XOR-malleable channel is simple because every modification of the transmitted message necessarily results in a different output. Hence, we use the simulator $\sigma_{\textbf{tls-mac}}$ that works as follows.

- On input the i-th message length $\ell_i \in \mathbb{N}$ at the inside interface, output a uniformly random $(\ell_i + 160)$-bit string $c_i \in_R \{0,1\}^{(\ell_i+160)}$ at the outside interface.

[1] TLS additionally allows the padding to extend the message by further blocks, although this is not widely implemented.

- On input the j-th message $\tilde{c}_j$ at the outside interface, if $\tilde{c}_j \neq c_j$ (or if c_j is not yet defined), then ignore all further messages input at the outside interface. Otherwise, input $\uparrow$ at the inside interface.

The validity of the construction then follows directly from the security of the MAC scheme.

Theorem 5.3. *Let* **tls-mac** $= (\mathsf{tls\text{-}tag}, \mathsf{tls\text{-}chk})$ *be the above described authentication protocol based on a MAC scheme* $\mathrm{MAC} = (\mathrm{mac}, \mathrm{check})$. *Then,* **tls-mac** *constructs from* $\multimap\!\!\oplus\!\!\twoheadrightarrow\!\bullet$ *and* $\bullet\!\!=\!\!=\!\bullet$ *the secure channel* $\bullet\!\!-\!\!\twoheadrightarrow\!\bullet$ *relative to the simulator* $\sigma_{\textsf{tls-mac}}$. *More formally,*

$$\left(\bullet\!\overset{k\text{-bit}}{=\!=}\!\bullet,\ \multimap\!\!\oplus\!\!\twoheadrightarrow\!\bullet\right) \xrightarrow{\textsf{tls-mac},\sigma_{\textsf{tls-mac}},\varepsilon} \bullet\!\!-\!\!\twoheadrightarrow\!\bullet,$$

where the symbols refer to sequences $\overset{q\text{ msgs}}{\multimap\!\!\oplus\!\!\twoheadrightarrow}\!\bullet$, $\overset{q\text{ msgs}}{\bullet\!\!-\!\!\twoheadrightarrow}\!\bullet$, *and* $\varepsilon_q = [\![\mathbf{G}_1^{\text{SUF-CMA}}(\mathrm{MAC})]\!] \circ (\cdot\mathbf{C}_{\textsf{mac}})$ *in* $q \in \{0, \ldots, 2^{64}-1\}$, *where the reduction converter* $\mathbf{C}_{\textsf{mac}}$ *is described in the proof.*

Proof. The availability condition means that

$$\textsf{tls-tag}^A\textsf{tls-chk}^B\left(\bullet\!\overset{k\text{-bit}}{=\!=}\!\bullet_\perp,\ \overset{q\text{ msgs}}{\multimap\!\!\oplus\!\!\twoheadrightarrow}\!\bullet_\perp\right) \quad\equiv\quad \overset{q\text{ msgs}}{\bullet\!\!-\!\!\twoheadrightarrow}\!\bullet_\perp,$$

for all $q \in \{0, \ldots, 2^{64}-1\}$. This follows immediately because tls-tag and tls-chk compute the MAC based on the same messages and keys, and hence the verification succeeds. Moreover, the MBO becomes 1 upon the same inputs in both cases.

The security condition means that

$$\left[\!\!\left[\left(\textsf{tls-tag}^A\textsf{tls-chk}^B\left(\bullet\!\overset{k\text{-bit}}{=\!=}\!\bullet,\ \overset{q\text{ msgs}}{\multimap\!\!\oplus\!\!\twoheadrightarrow}\!\bullet\right) \;\middle|\; \sigma_{\textsf{tls-mac}}{}^E\overset{q\text{ msgs}}{\bullet\!\!-\!\!\twoheadrightarrow}\!\bullet\right)\right]\!\!\right] \leq [\![\mathbf{G}_1^{\text{SUF-CMA}}(\mathrm{MAC})]\!] \circ (\cdot\mathbf{C}_{\textsf{mac}}),$$

for all $q \in \{0, \ldots, 2^{64}-1\}$. We use the notation

$$\mathbf{R}_q \coloneqq \textsf{tls-tag}^A\textsf{tls-chk}^B\left(\bullet\!\overset{k\text{-bit}}{=\!=}\!\bullet,\ \overset{q\text{ msgs}}{\multimap\!\!\oplus\!\!\twoheadrightarrow}\!\bullet\right) \quad\text{and}\quad \mathbf{S}_q \coloneqq \sigma_{\textsf{tls-mac}}{}^E\overset{q\text{ msgs}}{\bullet\!\!-\!\!\twoheadrightarrow}\!\bullet.$$

The setting considered in this proof is depicted in Figure 5.1. We define an MBO that captures that a MAC is accepted although the ciphertext was not forwarded unmodified, i.e., $E.i! \neq E.i?$ but $B.i! \neq \square$. Denoting this MBO as A, we have $\hat{\mathbf{R}}_q \models \mathbf{S}_q$, since the outputs at the E-interface are distributed identically (uniformly random bit strings of the same length) and $B.i! = A.i?$ whenever the MBO is not set.

The reduction $\mathbf{C}_{\mathsf{mac}}$, upon an input m at $A.i?$ outputs a uniformly random string $c_i \in \{0,1\}^{|m|+160}$ at $E.i!$. On input a value $E.i? \neq E.i!$ at the E-sub-interface, the reduction converter $\mathbf{C}_{\mathsf{mac}}$ queries $\mathbf{tag}(\mathrm{macEnc}(m_i, i-1))$ at $\mathbf{G}_1^{\text{SUF-CMA}}(\mathrm{MAC})$, computes the pair (m', t') similarly to $-\!\!\oplus\!\!\twoheadrightarrow\!\!\bullet$, and queries $\mathbf{vrf}\,(\mathrm{macEnc}(m', i-1), t)$ at $\mathbf{G}_1^{\text{SUF-CMA}}(\mathrm{MAC})$. We first observe that

$$\mathbf{R}_q \quad \overset{g}{\equiv} \quad \mathbf{C}_{\mathsf{mac}}\mathbf{G}_1^{\text{SUF-CMA}}(\mathrm{MAC}).$$

By the definition of the converter tls-chk, the MBO A is provoked only if the game $\mathbf{C}_{\mathsf{mac}}\mathbf{G}_1^{\text{SUF-CMA}}(\mathrm{MAC})$ is won. This concludes the proof. □

Our bound is different from the one proven by Krawczyk [Kra01] since we consider the one-time pad with a perfectly random key stream, we do not exhibit the quadratic "collision" term in the bound.

5.2.2 Construction Based on the Block-Malleable Channel

The authentication protocol used in TLS constructs a secure channel also from a block-malleable channel. The proof of this construction is slightly more complicated than for the case of the XOR-malleable channel, because a (non-trivial) modification at the channel can still result in the same plaintext message. The simulator $\sigma'_{\mathsf{tls\text{-}mac}}$ for this case initially samples a key $k \leftarrow\!\!\$\, \mathcal{K}$ and then proceeds as follows.

- On input the i-th message length $\ell_i \in \mathbb{N}$ at the inside interface, output a random $(\lceil \frac{\ell_i}{n} \rceil n + 160)$-bit string c_i with the same distribution as given by $-\!\!\boxed{n}\!\!\twoheadrightarrow\!\!\bullet$ at the outside interface.
- On input the j-th message $\tilde{c}_j$ at the outside interface, if $\tilde{c}_j \neq c_j$ (or if c_j is not yet defined), then ignore all further messages input at the outside interface. Otherwise, input $\uparrow$ at the inside interface.

The validity of the construction then follows directly from the security of the MAC scheme.

Theorem 5.4. *Let* **tls-mac** $=$ (tls-tag, tls-chk) *be the above described authentication protocol based on a MAC scheme* $\mathrm{MAC} = (\mathrm{mac}, \mathrm{check})$. *Then,* **tls-mac** *constructs from* $-\!\!\boxed{n}\!\!\twoheadrightarrow\!\!\bullet$ *and* $\bullet\!\!= =\!\!\bullet$ *the secure channel* $\bullet\!\!- -\!\!\twoheadrightarrow\!\!\bullet$. *More formally, for the simulator* $\sigma'_{\mathsf{tls\text{-}mac}}$,

$$(\bullet\!\!= =\!\!\bullet,\ -\!\!\boxed{n}\!\!\twoheadrightarrow\!\!\bullet) \quad \xRightarrow{\mathsf{tls\text{-}mac},\sigma'_{\mathsf{tls\text{-}mac}},\varepsilon} \quad \bullet\!\!- -\!\!\twoheadrightarrow\!\!\bullet,$$

where the symbols refer to sequences $-\!\!\boxed{n}\!\!\overset{q\ \text{msgs}}{\twoheadrightarrow}\!\!\bullet$ *and* $\bullet\!\!- \overset{q\ \text{msgs}}{-\!\!\twoheadrightarrow}\!\!\bullet$ *in* $q \in \{0, \ldots, 2^{64}-1\}$, *and performance measures* $\varepsilon_q = [\![\mathbf{G}_q^{\text{SUF-CMA}}(\mathrm{MAC})]\!] \circ (\cdot\mathbf{C}_{\mathsf{mac}}) - (q\ell/n)^2 2^{-n}$, *and the reduction* $\mathbf{C}_{\mathsf{mac}}$ *is described in the proof.*

As discussed in Section 5.1, the channel $\text{—}\boxed{n}\twoheadrightarrow\bullet$ is not a monotone system. The cases where this channel deviates from the monotone behavior can be captured by an MBO, and this MBO becomes 1 with probability at most $(q\ell/n)^2 2^{-n}$. This slightly reduces the performance of the reduction and is the reason for the exact definition of the performance in the reduction statement.

Proof. The availability condition means that

$$\mathsf{tls\text{-}tag}^A \mathsf{tls\text{-}chk}^B \left(\bullet\overset{k\text{-bit}}{\Leftarrow}\bullet_{\perp}, \text{—}\overset{q\text{ msgs}}{\boxed{n}}\twoheadrightarrow\bullet_{\perp} \right) \quad \equiv \quad \bullet\overset{q\text{ msgs}}{\text{—}}\twoheadrightarrow\bullet_{\perp}.$$

This follows immediately because tls-tag and tls-chk compute the MAC based on the same messages and keys, and hence the verification succeeds. Moreover, the both systems provide output until q messages have been input at the A-interface.

For the security condition, we have to show

$$\left[\!\!\left[\left(\mathsf{tls\text{-}tag}^A \mathsf{tls\text{-}chk}^B \left(\bullet\overset{k\text{-bit}}{\Leftarrow}\bullet, \text{—}\overset{q\text{ msgs}}{\boxed{n}}\twoheadrightarrow\bullet \right) \;\middle|\; \sigma'_{\mathsf{tls\text{-}mac}}{}^E \bullet\overset{q\text{ msgs}}{\text{—}}\twoheadrightarrow\bullet \right) \right]\!\!\right]$$
$$\leq \quad [\![\mathbf{G}_q^{\text{SUF-CMA}}(\text{MAC})]\!] \circ (\cdot\mathbf{C}_{\mathsf{mac}}).$$

We use the notation

$$\mathbf{R}_q \;\coloneqq\; \mathsf{tls\text{-}tag}^A \mathsf{tls\text{-}chk}^B \left(\bullet\overset{k\text{-bit}}{\Leftarrow}\bullet, \text{—}\overset{q\text{ msgs}}{\boxed{n}}\twoheadrightarrow\bullet \right) \quad \text{and} \quad \mathbf{S}_q \;\coloneqq\; \sigma'_{\mathsf{tls\text{-}mac}}{}^E \bullet\overset{q\text{ msgs}}{\text{—}}\twoheadrightarrow\bullet.$$

The setting considered in this proof is depicted in Figure 5.1. We define an MBO that captures that a MAC verification succeeds although the ciphertext was not forwarded unmodified. This can be defined on the system $\mathbf{R}_q$ as the condition "$E.i! \neq E.i?$ but $B.i! \neq \square$." On the system $\mathbf{S}_q$, this can be defined by adding a "hypothetical" computation of the MAC on input a message at the A-interface, and by performing the same computations as in $\text{—}\boxed{n}\twoheadrightarrow\bullet$ and tls-chk to check whether the verification succeeds. (The computation is purely hypothetical and is only used as a proof technique. All we need is that the event of a MAC forgery is defined.) Call this MBO A, we have

$$\hat{\mathbf{R}}_q \quad \overset{g}{\equiv} \quad \hat{\mathbf{S}}_q,$$

since the outputs at the E-interface are distributed identically (by the definition of the simulator) and $B.i! = A.i?$ whenever the MBO is not set. All that remains to be shown is that for all distinguishers

$$\left[\!\!\left[\hat{\mathbf{R}}_q \right]\!\!\right] \quad \leq \quad [\![\mathbf{G}_q^{\text{SUF-CMA}}(\text{MAC})]\!] \circ (\cdot\mathbf{C}_{\mathsf{mac}}),$$

for a reduction converter $\mathbf{C}_{\mathsf{mac}}$. This converter, upon input m at $A.i?$, queries $\mathbf{tag}(\mathrm{macEnc}(m, i-1))$ at the game $\mathbf{G}_q^{\text{SUF-CMA}}(\mathrm{MAC})$ and outputs a random string $c_i \in \{0,1\}^{(\lceil \frac{|m|}{n} \rceil n+160)}$ with the same distribution as in $\dashv\!\boxed{n}\!\twoheadrightarrow\!\bullet$ at $E.i!$. On input $E.j?$, the reduction behaves as follows.

- If $E.j? = E.j!$, then output $B.j! = A.j?$.
- Otherwise, reconstruct the message corresponding to the given ciphertext $E.j?$ analogously to $\dashv\!\boxed{n}\!\twoheadrightarrow\!\bullet$. That is, it is computed similarly to the CBC-mode decryption, where blocks used previously are treated as corresponding to the (first) previous occurrence of the ciphertext block, and fresh blocks are treated as corresponding to uniformly random plaintext blocks, parse the resulting plaintext into message $\tilde{m}$ and tag $\tilde{t}$. Query $\mathbf{vrf}((\mathrm{macEnc}(\tilde{m}, j), \tilde{t}))$ with the obtained pair at $\mathbf{G}_q^{\text{SUF-CMA}}(\mathrm{MAC})$.

Then, as $\hat{\mathbf{R}}_q \overset{g}{\equiv} \mathbf{C}_{\mathsf{mac}}\mathbf{G}_q^{\text{SUF-CMA}}(\mathrm{MAC})$, the theorem follows from Lemma 2.14. □

The bound differs slightly from the one given by Krawczyk [Kra01]. This is due to using a generic composition and to the more general modeling of the MAC: In Krawczyk's analysis, the block length of the cipher and the length of the MAC must be equal. In contrast, we do not restrict the size of the MAC.

Chapter 6

Conclusion

We introduced a framework that instantiates the paradigms of abstract and constructive cryptography of Maurer and Renner and allows to formalize and prove security statements about protocols in the setting of secure communication. The framework differs from previous approaches both in terms of the type of security statement and the underlying formal concepts. We showed in Chapter 4 how several construction steps can be achieved, performing an analysis of some well-known protocols, comparing the obtained security definitions to several security notions existing in the literature, and also discussing a new unilateral key-establishment protocol. In Chapter 5, we described the TLS record-layer protocol as two modular construction steps and analyzed them individually. The result verifies the soundness of this part of the TLS protocol in version 1.2. The goal of this chapter is to interpret the results of the thesis in a broader context.

Provable security and the constructive paradigm. Most widely deployed security protocols have not been designed with provable security in mind. The usefulness of provable security, at least in its current form, is even questioned not only by practitioners, but also by cryptography researchers [KM07]. One of the main criticisms can be exemplified using a line of works on ssh. The original ssh protocol was invented in 1995. Early versions of the protocol were not secure, but after several fixes, a version of the protocol could actually be (modified to become and) shown to be secure, which was proven by Bellare et al. [BKN04]. Somewhat surprisingly, Albrecht et al. [APW09] later showed a practical attack on the proven parts of the protocol. What had happened?

On a high level, the original analysis of Bellare et al. [BKN04] followed the "standard" attack-based security definitions and allowed the adversary

to submit ciphertexts to a "decryption oracle" which would then decrypt the given ciphertext using the secret key and, if the decryption succeeded, provide the adversary with the plaintext. In the realistic ssh protocol, however, an attacker can *fragment* ciphertexts and deliver them to the receiver in small chunks. The receiver decrypts the message once it recognizes that a complete "packet" has arrived. This mechanism allowed Albrecht et al. [APW09] to mount their attack: by using that the receiver would notice the end of the packet adaptively while processing the packet, they were able to obtain the information of *when* the receiver considered the packet to be complete, and this information leaks certain information about encrypted messages. In later work, Paterson and Watson [PW10] extended the provable security treatment of Bellare et al. [BKN04] to also cover this type of attack.

A somewhat related problem occurs in many practical security protocols (most of which were not designed by cryptographers), where cryptographic primitives and schemes are used in ways for which they were not specified and proven. The most impressive example in this respect is TLS, where, for instance, during the handshake neither of the key-establishment modes is chosen-ciphertext secure but the messages are sent without explicit authentication, the PRF for extracting from the key material is keyed with a value which is not guaranteed to be distributed uniformly, the keys generated during the key establishment are used for two different purposes, and the (not chosen-ciphertext secure) symmetric encryption schemes are used without authenticating the ciphertexts. Somewhat surprisingly, despite its ill-designed structure and after various vulnerabilities have been patched over time, the recent versions of the TLS protocol appear to achieve some reasonable level of security [JKSS12, KPW13, BFK+13a, BFK+13b, KMO+14]. Other examples include, for instance, ssh and Kerberos in their use of the symmetric encryption, or IPsec in using a PRF which is keyed with a publicly known nonce.

Several problems such as the implicit assumption that encryption also protects the integrity of messages and a wrong assignment of initialization vectors have occurred in various applications. One line of work in cryptography focuses on building definitions and schemes that are "harder to abuse:" authenticated encryption that protects both confidentiality and integrity of messages (e.g., [Rog02, RBB03]), nonce-based encryption in which the requirement on the IV to be uniformly random is replaced by the weaker requirement on the nonce to be unique [Rog04], and even *misuse-resistant authenticated encryption* [RS07] in which messages are protected as long as the *pair* of nonce and message is unique.

Both above problems hint to the problem that the guarantees implied by "standard" attack-based security proofs are not comprehensible from the security statements—at least not to practitioners making use of the schemes. In

other words, it was not clear what a scheme or protocol achieved when applied in a given scenario. This, however, appears to be a crucial requirement for provable security to be useful in practice: a security definition must specify the assumptions that the protocol makes in a way such that practitioners using the protocol can understand and verify whether the assumptions are met in their applications, as well as formalize the guarantees in a way that allows to verify whether these guarantees are sufficient for the intended application.

Constructive cryptography establishes a new approach to solving the described problems. One particular goal of constructive statements is that the semantics of security statements be clear by making all assumptions of a cryptographic scheme explicit in the assumed resources, and the guarantees in the constructed resource: if the encryption scheme does not protect the integrity of messages, then the construction will usually be stated as assuming an authenticated channel. Hence, this construction step can only be used in applications that guarantee that the ciphertext is not modified during the transmission. Ultimately, randomness should also be made explicit as a resource (by restricting to deterministic converters); in that case, the availability of a randomness for an IV and its assumed distribution would also appear as a resource in a construction statement. The hope is that if the assumptions of a scheme appear explicitly in the security statements and are comprehensible without understanding the details of the security model, then the abuse of schemes will decrease.

A consequent application of the constructive cryptography paradigm leads to proving modular construction steps in which each individual protocol is simple—consisting of a single method or scheme—and proven in isolation. This means that one ultimately builds a "library" of such construction steps, and complex protocols can be built by simply combining steps from the library. A protocol designer need not understand the exact security model in detail, because all steps that match syntactically can be composed. The composition theorem guarantees an overall protocol built from multiple construction steps is secure. The advantage of protocols built according to this approach is that they have a modular design and directly come with a security proof. We believe that the application of this paradigm in the design of future security protocols will help avoid practical vulnerabilities.

Abstraction and formalization. As most protocols or schemes considered in cryptography are discrete systems, security definitions for such protocols are stated with respect to a particular formal model of discrete systems. Various models of discrete systems appear in the cryptographic literature, most of these models are based on concepts such as Turing machines or automata. The main requirements for such a formal framework is that it has sufficient

expressiveness to capture the considered objects, that it is a sound abstraction of reality, and that it allows for comprehensible and precise proofs.

Most cryptographic definitions are phrased in terms of Turing machines; the motivation is that as most cryptographic schemes are only computationally secure, a model of computation is required to formulate and prove the security statements. Unfortunately, a Turing machine itself is a complex object, and the exact (mathematical) definition and properties of ITMs are hardly used, even in the fundamental theorems concerning the model. Indeed, it appears hardly tractable (and certainly not beneficial) to perform a full proof of a cryptographic protocol at such a low level; this level of detail is neither helpful nor necessary in the proofs, and ITMs are simply not the suitable abstraction. (This is also indicated in some of the works, such as by Hofheinz and Shoup [HS11].) Furthermore, the pseudo-code descriptions used to describe the Turing machines usually do not uniquely specify a Turing machine as a formal object, while they do, in contrast, usually specify a unique discrete system.

The top-down abstraction approach of Maurer and Renner [MR11] suggests to start from the opposite direction. Instead of beginning with a low-level model such as Turing machines and subsequently building more complex structures, one starts axiomatically from the high-level statements one is interested in and subsequently refines the concepts until the desired statements can be captured. In this spirit, we described discrete systems as an instantiation of the abstract system concept. The *type* of discrete systems captures exactly the properties required to prove statements about the behavior of systems and abstracts from, for example, concrete models of computation. The security of protocols can still be proved based on computational assumptions by showing *explicit reductions* that, intuitively, translate any attack on the cryptographic scheme into a solution for the underlying computational problem. These reductions are independent of a particular computational model and apply to any such model which satisfies the axioms of the system algebra. Following this approach allows us to make precise statements also about computationally secure protocol, without actually specifying the objects in a particular computational model.

The type of discrete system we describe in this thesis can be seen as a probabilistic variant of the type described by Kahn [Kah74] and restricts the type of behavior that can be captured; on a high level, it means that the behavior of a system may not depend on the *order* in which several messages arrive, only on their *values*. While most statements in this thesis have been made using only this type of system, some protocols such as Bracha's broadcast [Bra84] are inherently not of this type, and formalizing such a protocol requires a generalization of the discrete systems model. Due to the Brock-

Ackermann anomaly [BA83, Bro83], however, such a generalization is not straightforward; the obtained system algebra will not be connection-order invariant. This problem can be resolved by considering an algebra of sets of discrete systems, where each system in the set corresponds to one particular order in which the connections are applied—this can be seen as a type of scheduling. An algebra based on these sets still does not allow directly for modularization, but an approximating structure on the sets—roughly corresponding to the technique of Micciancio and Tessaro [MT13]—can be used to obtain modularity. We pursue this approach in ongoing work to define an extension of the model described in this thesis; the monotone discrete systems can then be seen as a "sub-algebra" of the more general type, allowing for simpler proofs.

Note that the model of discrete systems does not capture a notion of real time; that is, if a scheme leaks information because a computation for different inputs requires a different amount of time, the distinguisher in this model will not be able to observe the difference. This restriction applies to all current general cryptographic frameworks and can be resolved by modeling time explicitly on a lower abstraction layer.

Authentication and encryption. We have specified several natural types of communication channels and described several schemes and the constructions they achieve. A message authentication protocol based on a URF (which can be constructed by a PRF from a shared secret key) constructs an authenticated channel, and symmetric encryption schemes such as the one-time pad (this extends to stream ciphers because they construct a random key stream from a shared secret key) or CBC-mode encryption based on a URP (which can be constructed by a block cipher from a shared secret key) construct a fully secure channel in the authenticated setting. The natural composition of these two types of schemes then leads to the Encrypt-then-Authenticate transformation, the security follows immediately by the composition theorem.

We have also analyzed the Authenticate-then-Encrypt transformation by analyzing what encryption schemes achieve if the ciphertext is transmitted via an insecure channel; in the particular case of one-time pad encryption (again, this extends to stream ciphers) this is a confidential channel that allows the attacker to modify transmitted messages by applying an XOR mask for the plaintext message. We have then shown that a strongly unforgeable MAC constructs from this channel and a shared secret key a fully secure channel; hence, the Authenticate-then-Encrypt transformation is in principle also sound. In comparison with Encrypt-then-Authenticate, however, this analysis is less generic: while all authentication schemes construct the same authenticated channel from which the encryption scheme can start, the confidential

channels inherently depend on the exact encryption scheme that is employed, and hence the analysis for the MAC scheme has to be performed for each such channel independently.

The results shown in this thesis support the recommendation that Encrypt-then-Authenticate should be used for the composition of MACs and symmetric encryption schemes (see also, e.g., [Rog95, BN00]).

Relation to game-based notions. We have also shown how several notions described in the literature relate to constructions. In particular, a protocol based on a weakly unforgeable (WUF-CMA) MAC constructs an authenticated channel from an insecure channel and a shared secret key. A protocol based on an encryption scheme constructs a secure channel from an authenticated channel and a key if and only if the underlying scheme satisfies the IND-CPA property. Also, such a protocol constructs a confidential and non-malleable channel from an insecure channel and a key if the scheme is IND-CCA, and a secure channel from an insecure one and a key if the scheme satisfies both IND-CPA and INT-CTXT. Additionally, we have shown that an authentication protocol based on an existentially unforgeable (EUF-CMA) signature scheme constructs a multiple-use authenticated channel from a single-use authenticated channel. (The construction is phrased in a scenario with multiple receivers, but of course also applies to the three-party setting.)

We view these statements as confirming the soundness of the respective game-based notions. Note that for cryptographic schemes in the "traditional" sense, it is not *a priori* clear how the composition of multiple schemes should be defined; this might be the reason for "obscure" combinations of schemes such as Authenticate-and-Encrypt like in ssh. Converters, in contrast, assign all inputs and outputs explicitly to the input and output interfaces, and the composition of two converters fully describes the combined converter that one obtains.

Some games such as the original formulation of INT-PTXT do not have a constructive interpretation because they inherently do not allow for composable statements, this has been discussed by Maurer et al. [MRT12]. Other games, such as NM-CPA or IND-CCA1, appear to correspond to constructions between unnatural resources [CMT13a]. These results question the justification for the corresponding game-based notions, but do not exclude the usefulness in specific applications.

Anonymity. In Section 4.6, we have shown how statements about anonymous communication can be phrased constructively, providing the (to the best of our knowledge) first treatment of anonymity within a composable security framework. The statement makes explicit in which sense anonymity

can be achieved by specifying a set of potential receivers relative to which the anonymity holds. We have then shown that a public-key encryption scheme that fulfills the properties introduced by Bellare et al. [BBDP01] and Abdalla et al. [ABN10] constructs a *receiver-anonymous* confidential channel from a *receiver-anonymous* insecure channel and authenticated channels from the receivers to the sender. In particular, we showed that confidentiality, key privacy, and weak robustness are indeed sufficient for such a scheme to be useful. Our results do not only support the trust in existing schemes and constructions; they also show that the simpler and more efficient weakly robust schemes (see [ABN10]) can be used safely. The same approach to defining anonymity has been followed in the subsequent work of Alwen et al. [AHM+14]. An interesting question for future work is whether the guarantees achieved by overlay networks such as the TOR protocol can also be explicitly phrased constructively following this approach.

Unilateral key establishment. We have presented a simple and efficient protocol for unilateral key establishment, which can be viewed as a modularization and generalization of the protocol proposed by Shoup [Sho99]. The setting with unilateral authentication occurs naturally in the client-server scenario that is present in most Internet protocols, where only the server has a certified public key. Despite the practical importance of this scenario (if the client does not have a certified public key, a mutually authenticated key cannot be achieved), the majority of models and protocols in the cryptographic literature focuses on the mutual-authentication case, where both the client and the server have certified public keys. Moreover, all previous security definitions that *do* apply to this case come (*a priori*) without composition guarantees.

The protocol we describe is built around a key-encapsulation mechanism that is only CPA-secure, while previous such constructions generally required the KEM to be CCA-secure (see, e.g., [MSW08, CMT13a]). This is achieved by using the KEM in the opposite direction: the public key is generated by the client and transmitted over an insecure channel, and the ciphertext encapsulating the key is sent authentically. Appending the public key to the ciphertext sent over the authenticated connection allows the client to check that it was not modified during the transmission. Indeed, it appears beneficial to replace the key-establishment step in existing protocols such as TLS by simpler and more efficient protocols based on the idea presented in Section 4.7.

The guarantee formalized by the unilateral key is indeed useful: mutual authentication can be achieved using a (pre-shared low-entropy) password. As shown by Maurer et al. [MTC13], one can alternatively apply, e.g., authenticated encryption to obtain a multiple-message channel which offers “partner

invariance:" either the authenticated party (here B) communicates consistently with A, or B communicates consistently with E, and send the password via this channel.

We also showed how the authenticated channel assumed by the construction can itself be constructed from realistic assumptions (such as a PKI) by a protocol based on nonces and signatures. This assumption is modeled as a single-use authenticated channel $\succ\!\!\diamond\!\!-\!\!\bullet$ that allows the server to transmit a single message (the signature verification key) to all clients authentically is available. This channel can itself be constructed in several ways, such as by pre-installing the verification key at all clients (this is the assumption used in previous models) or by use of a public-key infrastructure with a pre-installed public key of a certification authority at the clients (this is the usual implementation in practice); our protocol is independent of how this channel is constructed. The analysis of global public-key infrastructures in general settings, i.e., with respect to corruptions, will require a generalization of the described constructions in a similar sense as done by Canetti et al. [CSV14] in the context of the UC framework. This, as a topic of independent interest, will be discussed in future work.

SSL/TLS record layer. The SSL/TLS protocol has received considerable attention in the cryptographic literature recently because of its importance in practice. Unfortunately, the protocol was not designed with provable security in mind, and multiple severe vulnerabilities have been detected and patched in recent years. Several of these vulnerabilities also affected the record-layer protocol, such as an incorrect chaining in CBC mode or an attack that became possible due to verbose error messages. The proof of the TLS record-layer protocol presented in Chapter 5 is based on the first proof of the Authenticate-then-Encrypt transformation that was applicable to the parameters in TLS [MT10] (the previous work by Krawczyk [Kra01] analyzed similar schemes, but the parameters were chosen such that they did not apply to TLS, and padding was ignored).

While the proof (and also the subsequent work by Paterson et al. [PRS11]) shows that the specification of the record layer in TLS version 1.2 [DR08] does not exhibit further vulnerabilities, it still seems beneficial to replace the Authenticate-then-Encrypt transformation by Encrypt-then-Authenticate (or authenticated encryption) as AtE is inherently susceptible to implementation flaws and timing attacks. While the Binary Packet Protocol (BPP) in ssh, based on Encrypt-and-Authenticate, has also suffered from vulnerabilities [APW09], the design of the Encapsulating Security Payload (ESP) in IPsec demonstrates that an EtA approach indeed leads to a sound protocol design, as shown by Jost [Jos14].

Bibliography

[Aba02] Martín Abadi. Private authentication. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 27–40. Springer, 2002.

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *Theory of Cryptography*, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497. Springer, 2010.

[ABP+13] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. On the security of RC4 in TLS and WPA. In *USENIX Security Symposium*, 2013.

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[AF04] Martín Abadi and Cédric Fournet. Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, 2004.

[AHM+14] Joël Alwen, Martin Hirt, Ueli Maurer, Arpita Patra, and Pavel Raykov. Anonymous authentication with shared secrets. Cryptology ePrint Archive, Report 2014/073, February 2014. `http://eprint.iacr.org/2014/073`.

[AP12] Nadhem J. AlFardan and Kenneth G. Paterson. Plaintext-recovery attacks against datagram TLS. In *Network and Distributed System Security Symposium (NDSS'12)*, 2012.

[AP13] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *IEEE Symposium on Security and Privacy*, 2013.

[APW09] Martin A. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext recorvery attacks against SSH. In *IEEE Symposium on Security and Privacy*, pages 16–26. IEEE, 2009.

[BA83] J. Dean Brock and William B. Ackermann. Scenarios: A model of non-determinate computation. In *Formalization of Programming Concepts*, 1983.

[Bar04] Gregory V. Bard. Vulnerability of SSL to chosen-plaintext attack. Cryptology ePrint Archive: Report 2004/111, May 2004.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology — Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 419–428. ACM, 1998.

[BD03] Amos Beimel and Shlomi Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16(1):25–39, 2003.

[BDJR97] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, pages 394–403. IEEE, 1997.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[Bea91] Donald Beaver. Foundations of secure interactive computing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391. IACR, Springer, 1991.

[BFCZ12] Karthikeyan Bhargavan, Cédric Fournet, Ricardo Corin, and Eugen Zălinescu. Verified cryptographic implementations for TLS. In *ACM Transactions on Information and System Security (TISSEC'12), volume 15(1): 3*, 2012.

[BFK+13a] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, and Pierre-Yves Strub. Implementing TLS with verified cryptopgrahic security. In *IEEE Symposium on Security and Privacy*, pages 445–469, 2013.

[BFK+13b] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Santiago Zanella-Béguelin. Proving the TLS handshake secure (as it is). Technical report, 2013.

[BFS+13] Christina Brzuska, Marc Fischlin, Nigel Smart, Bogdan Warinschi, and Steve Williams. Less is more: Relaxed yet composable security notions for key exchange. *International Journal of Information Security*, 12(4):267–297, 2013.

[BFWW11] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of Bellare-Rogaway key exchange protocols. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 51–62. ACM, ACM Press, 2011.

[BGM04] Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309, November 2004.

[BJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key exchange protocols and their security analysis. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, 1997.

[BKN04] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):206–241, 2004.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, December 2000.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Sciece*, pages 1–12. Springer, 1998.

[BM98] Simon Blake-Wilson and Alfred Menezes. Entity authentication and key transport protocols employing asymmetric techniques. In Stafford Tavares and Henk Meijer, editors, *Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*. Springer, 1998.

[BM03] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2003.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000. Journal version in [BN08].

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008.

[BPW07] Michael Backes, Birgit Pfitzmann, and Michael Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12):1685–1720, December 2007.

[BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.

[BR00] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, 2000.

[BR06] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. In Serge Vaudenay, editor, *Advances in Cryptology — EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[Bra84] Gabriel Bracha. An asynchronous $[(n-1)/3]$-resilient consensus protocol. In *ACM Symposium on Principles of Distributed Computing*, pages 154–162, 1984.

[Bro83] J. Dean Brock. *A Formal Model of Non-determinate Dataflow Computation*. PhD thesis, Massachusetts Institute of Technology, 1983.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001. Extended version in [Can13].

[Can13] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, July 2013. Original version from December 2000.

[CCK+06] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-structured probabilistic I/O automata. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 207–214. IEEE, 2006.

[CCLP07] Ran Canetti, Ling Cheung, Nancy Lynch, and Oliver Pereira. On the role of scheduling in simulation-based security. 7th International Workshop on Issues in the Theory of Security, March 2007.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[CK02a] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 27–52. Springer, 2002.

[CK02b] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 3027 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2002.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[CMT13a] Sandro Coretti, Ueli Maurer, and Björn Tackmann. Constructing confidential channels from authenticated channels—Public-key encryption revisited. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology — ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 134–153. Springer, 2013.

[CMT13b] Sandro Coretti, Ueli Maurer, and Björn Tackmann. *A Constructive Perspective on Key Encapsulation*, volume 8260 of *Lecture Notes in Computer Science*, chapter "Security and Privacy", pages 226–239. Springer, August 2013.

[CSV14] Ran Canetti, Daniel Shahaf, and Margarita Vald. Composable authentication with global PKI. Cryptology ePrint Archive Report 2014/432, June 2014. `http://eprint.iacr.org/2014/432`.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DF13] Yevgeniy Dodis and Dario Fiore. Interactive encryption, message authentication, and anonymous key exchange. Cryptology ePrint Archive, Report 2013/817, December 2013. `http://eprint.iacr.org/2013/817`.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[DR08] Tim Dierks and Eric Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246, August 2008.

[DR11] Thai Duong and Juliano Rizzo. Here come the XOR ninjas. In *Ekoparty*, 2011.

[FHM+12] Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben, and Matthew Smith. Why Eve and Mallory love Android: An analysis of Android SSL (in)security. In *Proceedings of the ACM Conference on Computer and Communications Security (ACM CCS'12)*, pages 50–61, 2012.

[Fis07] Marc Fischlin. Anonymous signatures made easy. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public-Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 31–42. Springer, 2007.

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 352–368. Springer, 2013.

[FS03] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. Wiley, 2003.

[GDK02] Virgil D. Gligor, Pompiliu Donescu, and Jonathan Katz. On message integrity in symmetric encryption. February 2002.

[GIJ+12] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. The most dangerous code in the world: Validating SSL certificates in non-browser software. In *Proceedings of the ACM Conference on Computer and Communications Security (ACM CCS'12)*, pages 38–49, 2012.

[GKS13] Florian Giesen, Florian Kohlar, and Douglas Stebila. On the security of tls renegotiation. In *ACM Conference on Computer and Communications Security*, pages 387–398, 2013.

[GL90] Shafi Goldwasser and Leonid Levin. Fair computation of general functions in presence of immoral majority. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology — CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 1990.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. Earlier version in *STOC 1982*.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of*

the 17th Annual ACM Symposium on Theory of Computing, pages 281–304. ACM Press, 1985.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game—a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM, 1987.

[GSU13] Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. Anonymity and one-way authentication in key exchange protocol. *Designs, Codes and Cryptography*, 67(2):245–269, May 2013.

[Hic95] Kipp E.B. Hickman. The SSL protocol. February 1995.

[HK99] Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and System Security (TISSEC)*, 2(3):230–268, August 1999.

[HM08] Alejandro Hevia and Daniele Micciancio. An indistinguishability-based characterization of anonymous channels. In Nikita Borisov and Ian Goldberg, editors, *Privacy Enhancing Technologies*, volume 5134 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2008.

[HMS+13] Thomas Holenstein, Ueli Maurer, Angelika Steger, Emo Welzl, and Peter Widmayer. Algorithms, probability, and computing. Lecture Notes, ETH Zürich, September 2013.

[HMU05] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Polynomial runtime in security definitions. In *18th IEEE Workshop on Computer Security Foundations*, pages 156–169. IEEE, 2005.

[HS11] Dennis Hofheinz and Victor Shoup. GNUC: A new universal composability framework. Cryptology ePrint Archive, Report 2011/303, June 2011.

[HZ10] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henry Gilbert, editor, *Advances in Cryptology — EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 466–485. Springer, 2010.

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *Proceedings of the 47th Symposium on Foundations of Computer Science*, pages 239–248, 2006.

[Iwa06] Tetsu Iwata. New blockcipher modes of operation with beyond the birthday bound security. In Matthew Robshaw, editor, *Fast Software Encryption*, volume 4047 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2006.

[JK02] Jakob Jonsson and Burton S. Kaliski Jr. On the security of RSA encryption in TLS. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 127–142. Springer, 2002.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology — CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 273–293. Springer, 2012.

[Jos14] Daniel Jost. A constructive analysis of IPsec. Master's thesis, ETH Zurich, April 2014.

[Kah74] Gilles Kahn. The semantics of a simple language for parallel programming. In Jack L. Rosenfeld, editor, *Information Processing 74*, pages 471–475. North-Holland, 1974.

[Kel02] John Kelsey. Compression and information leakage of plaintext. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption*, volume 2365 of *Lecture Notes in Computer Sciencec*, pages 263–276. Springer, 2002.

[KM07] Neal Koblitz and Alfred Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007.

[KMO+13] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Björn Tackmann, and Daniele Venturi. Anonymity-preserving public-key encryption: A constructive approach. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies*, volume 7981 of *LNCS*, pages 19–39. Springer, 2013.

[KMO+14] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Björn Tackmann, and Daniele Venturi. (De-)Constructing TLS. Cryptology ePrint Archive, Report 2014/020, January 2014. `http://eprint.iacr.org/2014/020`.

[KPR03] Vlastimil Klíma, Ondrej Pokorný, and Tomá Rosa. Attacking RSA-based sessions in SSL/TLS. In Colin D. Walter, Çetin K. Koç,

and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 426–440. Springer, 2003.

[KPW13] Hugo Krawcyzk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan Garay, editors, *Advances in Cryptology — CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448. Springer, 2013.

[Kra01] Hugo Krawczyk. The order of encryption and authentication for protecting communications. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.

[Kro99] Maxwell Krohn. On the definitions of cryptographic security: Chosen-ciphertext attack revisited. Senior Thesis, Harvard University, 1999.

[KSS13] Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DH and TLS-RSA in the standard model. Cryptology ePrint Archive, Report 2013/367, June 2013.

[KT09] Ralf Küsters and Max Tüngerthal. Universally composable symmetric encryption. In *In Proceedings of the 22nd IEEE Computer Security Foundations Symposium*, pages 293–307. IEEE, 2009.

[KT13] Ralf Küsters and Max Tüngerthal. The IITM model: A simple and expressive model for universal composability. Cryptology ePrint Archive, Report 2013/025, January 2013.

[Küs06] Ralf Küsters. Simulation-based security with inexhaustible interactive Turing machines. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop*, pages 309–320. IEEE, 2006.

[KY00a] Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *Proceedings of the 32nd annual ACM Symposium on Theory of Computing*, pages 245–254. ACM, 2000.

[KY00b] Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In Bruce Schneier, editor, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer, 2000.

[Lan10] A. Langley. Transport layer security snap start, June 2010.

[LLM07] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi My, editors, *ProvSec 2007*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2007.

[LMM10] A. Langley, N. Modadugu, and B. Moeller. Transport layer security false start, June 2010.

[Mau02] Ueli Maurer. Indistinguishability of random systems. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–132. Springer, 2002.

[Mau11] Ueli Maurer. Constructive cryptography: A new paradigm for security definitions and proofs. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *TOSCA 2011—Theory of Security and Applications*, volume 6993 of *Lecture Notes in Computer Science*, pages 33–56. Springer, 2011.

[Mau13] Ueli Maurer. Conditional equivalence of random systems and indistinguishability proofs. In *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 3150–3154, July 2013.

[Mau14] Ueli Maurer. Cryptography. Lecture Notes, ETH Zürich, February 2014.

[Mey14] Christopher Meyer. *20 Years of SSL/TLS Research—An Analysis of the Internet's Security Foundation*. PhD thesis, Ruhr Universität Bochum, Bochum, February 2014.

[Moh10] Payman Mohassel. A closer look at anonymity and robustness in encryption schemes. In Masayuki Abe, editor, *Advances in Cryptology — ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 501–518. Springer, 2010.

[MPR07] Ueli Maurer, Krzysztof Pietrzak, and Renato Renner. Indistinguishability amplification. In Alfred Menezes, editor, *Advances in Cryptology — CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 130–149. Springer, August 2007.

[MR91] Silvio Micali and Phillip Rogaway. Secure computation. In Joan Feigenbaum, editor, *Advances in Cryptology—CRYPTO 91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 1991.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In *Innovations in Computer Science*. Tsinghua University Press, 2011.

[MRT12] Ueli Maurer, Andreas Rüedlinger, and Björn Tackmann. Confidentiality and integrity: A constructive perspective. In Ronald Cramer, editor, *Theory of Cryptography — TCC 2012*, volume 7194 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 2012.

[MS96] Ueli Maurer and Pierre Schmid. A calculus for security bootstrapping in distributed systems. *Journal of Computer Security*, 4(1):55–80, 1996.

[MSW08] Paul Morrissey, Nigel Smart, and Bogdan Warinschi. A modular security analysis of the TLS handshake protocol. In Josef Pieprzyk, editor, *Advances in Cryptology — ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 55–73. Springer, 2008.

[MT10] 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*. ACM, 2010.

[MT13] Daniele Micciancio and Stefano Tessaro. An equational approach to secure multi-party computation. In *Innovations in Computer Science*, 2013.

[MTC13] Ueli Maurer, Björn Tackmann, and Sandro Coretti. Key exchange with unilateral authentication: Composable security definition and modular protocol design. Cryptology ePrint Archive Report 2013/555, September 2013.

[Nam02] Chanathip Namprempre. Secure channels based on authenticated encryption schemes: A simple characterization. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 111–118. Springer, 2002.

[NMO08] Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto. Relationship of three cryptographic channels in the UC framework. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec 2008*, volume 5324 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 2008.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd annual ACM Symposium on Theory of Computing*, pages 427–437. ACM, 1990.

[PRS11] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology — ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 372–389. Springer, 2011.

[PW85] Andreas Pfitzmann and Michael Waidner. Networks without user observability. In Franz Pichler, editor, *Advances in Cryptology — EUROCRYPT '85*, volume 219 of *Lecture Notes in Computer Science*, pages 245–253. Springer, 1985.

[PW01] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 184–200. IEEE, 2001.

[PW10] Kenneth G. Paterson and Gaven J. Watson. Plaintext-dependent decryption: A formal security treatment of SSH-CTR. In Henry Gilbert, editor, *Advances in Cryptology — EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2010.

[RBB03] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient symmetric encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3):365–403, August 2003.

[RCC+11] S. Radhakrishnan, Y. Cheng, J. Chu, J. Jain, and B. Raghavan. TCP fast open. In *Conference on Emerging Networking Experiments and Technologies*, pages 21:1–21:12. ACM, 2011.

[RD09] Marsh Ray and Steve Dispensa. Renegotiating TLS. November 2009.

[Ren05] Renato Renner. *Security of Quantum Key Distribution*. PhD thesis, ETH Zürich, September 2005.

[Rog95] Phillip Rogaway. Problems with proposed IP cryptography. Draft, April 1995. http://www.cs.ucdavis.edu/˜rogaway/papers/draft-rogaway-ipsec-comments-00.txt.

[Rog02] Phillip Rogaway. Authenticated encryption with associated data. In *ACM Conference on Computer and Communications Security*, pages 98–107. ACM Press, 2002.

[Rog04] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2004.

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology — EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006. Full version appeared as [RS07].

[RS07] Phillip Rogaway and Thomas Shrimpton. Deterministic authenticated encryption: A provable-security treatment of the key-wrap problem. Cryptology ePrint Archive, Report 2006/221, August 2007. Full version of [RS06].

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, October 1949.

[SHI+12] E. Stark, L.-S. Huang, D. Israni, C. Jackson, and D. Boneh. The case of prefetching and prevalidating TLS server certificates. In *Proceedings of the Symposium on Network and Distributed Systems Security*. Internet Society, 2012.

[Sho99] Victor Shoup. On formal models for secure key exchange. Research Report RZ 3120, IBM, April 1999.

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001.

[Shr04] Tom Shrimpton. A characterization of authenticated-encryption as a form of chosen-ciphertext security. Cryptology ePrint Archive, Report 2004/272, October 2004.

[SL94] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. In Bengt Jonsson and Joachim Parrow, editors, *Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*, pages 481–496. Springer, 1994.

[Vau02] Serge Vaudenay. Security flaws induced by CBC padding — applications to SSL, IPSEC, WTLS... In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–545. Springer, 2002.

[WFS03] Brent R. Waters, Edward W. Felten, and Amit Sahai. Receiver anonymity via incomparable public keys. In *ACM Conference on Computer and Communications Security*, pages 112–121. ACM, 2003.

[WHF02] Doug Whiting, Russ Housley, and Niels Ferguson. Counter with CBC-MAC (CCM). Submission to NIST, June 2002.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE, 1982.

[YWDW06] Guimin Yang, Duncan S. Wong, Xiaotie Deng, and Hauxiong Wang. Anonymous signature schemes. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography — PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 347–363. Springer, 2006.

[ZS92] Yuliang Zheng and Jennifer Seberry. Practical approaches to attaining security against adaptively chosen ciphertext attacks (extended abstract). In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO 1992*, volume 1440 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 1992.