Diss. ETH No. 26723

On Generalizations of Composable Security

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

presented by

Daniel Jost
MSc ETH in Computer Science, ETH Zurich

born on February 26, 1989
citizen of Eriswil BE, Switzerland

accepted on the recommendation of

Prof. Dr. Ueli Maurer, examiner
Prof. Dr. Dennis Hofheinz, co-examiner
Prof. Dr. Jesper Buus Nielsen, co-examiner

2020

Daniel Jost
ORCID: 0000-0002-6562-9665

DOI: 10.3929/ethz-b-000417544

Acknowledgments

First and foremost, I would like to thank my advisor Ueli Maurer for the opportunity to do a PhD in the fascinating field of cryptography. His passion for research and quest for finding the right answers to fundamental problem in cryptography have deeply inspired me and shaped my interests in research. He always had time for me and his unique way of thinking led to many fruitful discussions.

Sincere thanks go to Dennis Hofheinz and Jesper Buus Nielsen for willing to co-referee this thesis and their valuable feedback.

For the countless interesting discussions and all the good times we have had together, I would like to take the opportunity to thank all the current and former members of the Information Security and Cryptography Research Group: Joël Alwen, Christian Badertscher, Fabio Banfi, Sandro Coretti, Grégory Demay, Robert Enderlein, Martin Hirt, David Lanzenberger, Chen-Da Liu Zhang, Christian Matt, Marta Mularczyk, Christopher Portmann, Pavel Raykov, Guilherme Rito, Gregor Seiler, Björn Tackmann, Daniel Tschudi, Jiamin Zhu, and Vassilis Zikas. It would not have been the same without you all!

I would like to express special gratitude to all my co-authors during my PhD studies—Chen-Da, Christian B., Christian M., and Marta—for the exciting former and hopefully future collaborations and the fruitful exchange of ideas. Moreover, I thank Beate Bernhard and Claudia Günthart for their administrative support.

Last but not least, this thesis would not have been possible without the invaluable encouragement of my family, and the support of my dear friends.

Abstract

Cryptography is a cornerstone for the protection of the digital society, and security definitions lie at the heart of cryptographic research. Their importance stems from the fact that for a cryptographic scheme, in contract to its correctness, security cannot simply be demonstrated. Rather, the widely accepted paradigm is to build confidence by mathematically proving their security. This foremost requires to have a sound and meaningful mathematical security model, such that the implied guarantees actually correspond to the intended properties.

The paradigm of provable security can be traced back to Shannon's seminal work on the one-time pad encryption. His custom-made and purely information-theoretic definition, however, does not handily generalize to other types of schemes and flavors of security. Nowadays, most security definitions are game-based, meaning that a protocol is secure if an attacker is unable to perform a certain set of attacks in a prescribed and simplified interaction with the scheme. This, however, has severe drawbacks. First, the match between the formal security definition and the intended use-case of the corresponding application is not performed rigorously, but typically only argued informally. Second, such security definitions do not compose. For instance, a provably secure symmetric encryption scheme might turned out insecure when combined with a provably secure key-agreement scheme, due to the lack of composability of the respective definitions.

A in many respects superior approach are composable (sometimes called simulation-based) security frameworks. In such a framework, the goal of a cryptographic primitive can be seen as providing a construction of a so-called ideal resource from an assume resource, for a well-defined construction notion. The ideal resource thereby formalizes what must be

achieved—in any possible context—in a clean and abstract manner. For instance, the goal of an encryption scheme can be seen as construction of a secure channel that does not leak the messages, but at most their length, to an eavesdropper, hence making the achieved guarantees explicit rather than listing excluded attacks. The assumed resource, on the other hand, makes explicit what is required to be available to the involved parties, e.g., a shared secret key and an authentication channel. Moreover, by design, those frameworks ensure that the security of an overall scheme is directly implied by the security of its components, making them well suited for modular protocol designs and analyses.

However, up to now, the adoption of composable security statements has been hindered by a number of obstacles. In particular, the imposed abstraction boundaries lack flexibility, impeding modularity as it prevents certain resources from being decomposed. Moreover, the traditional types of composable security statements are frequently overly specific, e.g., encoding a very particular assumed or constructed resource, hindering their reuse. Some popular abstraction boundaries, such as the random oracle, are even known to be outright impossible to construct from resources readily available in the real world. In addition, composable frameworks are also met with some skepticism because of many impossibility results; goals such as commitments and zero-knowledge that are achievable in a stand-alone sense were shown to be unachievable composably (without a setup). In particular, in the context of adaptive security, the so-called "simulator commitment problem" arises: once a party gets corrupted, an efficient simulator is unable to be consistent with its pre-corruption outputs.

Generalizing Constructive Cryptography. In this thesis, we generalize and extend the Constructive Cryptography framework in three ways to overcome those limitations.

First, we propose the extension of the Constructive Cryptography framework by so-called *global event histories*, to enable more flexible abstraction boundaries. That is, in addition to the traditional input-output behavior, each resource can trigger events from a predefined set. Other resources can then depend on the global event history, i.e., on which events having occurred so far and in which order. For example, a key resource (one module) can now export the event of having been leaked, upon which a channel (another, apparently independent module)

can degrade its security guarantees.

Second, we propose the notion of *context-restricted constructions*. Context-restricted constructions allow us to model that certain resources can only be used in an explicitly specified set of contexts. Such a notion with explicitly limited composition guarantees is particularly useful when dealing with abstractions such as a random oracle, where no weaker and feasible to instantiate, yet useful abstraction boundary are known. As an application, we then show that the notion of universal computational extractors (UCE), introduced by Bellare et al. as an alternative to the random oracle model, can be understood as a special case of context-restricted constructions.

Third, we revisit the so-called *simulator commitment problem*, that mainly occurs in the context of adaptive security. We introduce a novel composable security notion that is able to express the composable guarantees of schemes that previously only admitted standalone security definitions, while providing clean semantics of how the guarantees should be interpreted, holding in any environment, and being equipped with a composition theorem. In a nutshell, we introduce *interval-wise* guarantees formalizing properties that hold in between two events, not forcing the simulation to be consistent between the intervals (the root of the commitment issues). On a technical level, we leverage the specification-based approach of the CC framework, where proving a protocol $\boldsymbol{\pi}$ to be secure corresponds to modeling the assumed real-world specification $\mathcal{R}$, and showing that $\boldsymbol{\pi}\mathcal{R}$ is contained in an ideal specification $\mathcal{S}$. We formalize interval-wise guarantees as a novel type of specifications and develop the required theory.

A case study. Finally, we consider two-party *secure messaging*, proposing the first systematic and composable security model thereof. Taking advantage of our notion of global events, we develop a unified and easily reusable type security statement for each of the involved sub-protocol. This is in stark contrast to existing game-based security treatments, where the security games of sub-protocols are tailored to the overall protocol analysis, impeding their reuse in other protocols and contexts. Furthermore, we utilize our novel types of specifications to deal with the commitment problem, naturally occurring in secure messaging where one considers adaptive exposure of parties' secret state.

Zusammenfassung

Die Kryptographie ist ein Eckpfeiler im Schutze der digitalen Gesellschaft und Sicherheitsdefinitionen spielen eine zentrale Rolle in der kryptographischen Forschung. Ihre Wichtigkeit beruht auf der Tatsache, dass für eine kryptographisches Schema Sicherheit, im Gegensatz zu Korrektheit, nicht empirisch demonstriert werden kann. Stattdessen ist es der breit akzeptierte Ansatz, Vertrauen durch einen mathematischen Sicherheitsbeweis zu bilden. Dies erfordert in erster Linie jedoch die Existenz eines sinnhaften mathematischen Sicherheitsmodells, für welches die implizierten Garantien effektiv die gewünschten Sicherheitseigenschaften implizieren.

Das Paradigma der beweisbaren Sicherheit kann bis zu Shannons einflussreicher Arbeit über das One-Time-Pad Schema zurückverfolgt werden. Jedoch lässt seine massgeschneiderte und rein informationstheoretische Sicherheitsdefinition nicht einfach auf andere Sicherheitsvarianten verallgemeinern. Heutzutage sind die meisten Sicherheitsdefinitionen durch ein Spiel zwischen einem Angreifer und einem Herausforderer definiert. Das heisst, ein Schema wird sicher genannt, wenn kein Angreifer in dieser vereinfachten Interaktion mit dem Schema eine Attacke durchführen kann. Dies hat jedoch einige schwerwiegende Nachteile. Ersten wird die Korrespondenz zwischen der formalen Sicherheitsdefinition und der beabsichtigen Anwendung oft nicht gründlich durchgeführt, sondern oft nur informell argumentiert. Zweitens lassen sich solche Sicherheitsdefinitionen nicht zusammensetzen. So kann es z.B. sein, dass ein beweisbar sicheres symmetrisches Verschlüsselungsprotokoll sich als unsicher hersausstellt, sobald es mit einem beweisbar sicheren Schlüsseltauschprotokoll kombiniert wird, weil die entsprechenden Definitionen nicht zusammensetzbar sind.

Ein in vielerlei Hinsicht besserer Ansatz sind sogenannte *composable* (zusammensetzbare) Sicherheits-Frameworks, welche teilweise auch simulationsbasiert genannt werden. In einem solchen Framework kann das Ziel einer kryptographischen Primitive kann dann als die Konstruktion einer sogenannten idealen Ressource aus einer angenommen Ressource (für einen wohldefinierten Begriff der Konstruktion) gesehen werden. Die ideale Ressource formalisiert dabei das in jedem Kontext zu erreichende Ziel auf eine simple und abstrakte Art. So kann zum Beispiel das Ziel eines Verschlüsselungsverfahren in der Konstruktion eines sicheren Kommunikationskanals, welcher höchstens die Nachrichtenlänge einem Mitlauscher preisgibt, gesehen werden. Folglich werden die erreichten Garantien explizit gemacht, statt ausgeschlossene Attacken aufgelistet. Auf der anderen Seite machen die angenommenen Ressourcen die Voraussetzungen, wie z.B. einen geteilten sicheren Schlüssel oder einen authentischen Kommunikationskanal, an die involvierten Parteien explizit. Zusätzlich sind die entsprechenden Frameworks so entworfen, dass sich die Sicherheit des Gesamtprotokolls direkt aus der Sicherheit der Komponenten ergibt, wodurch sie besonders gut für den modularen Protokollentwurf und die modulare Analyse geeignet sind.

Jedoch ist bis heute die Adoption von composable Sicherheitsaussagen durch eine Anzahl Hemmnisse erschwert. Insbesondere lassen die dadurch auferlegten Abstraktionsschnittstellen an Flexibilität vermissen, wodurch sich gewisse Ressourcen nicht modular zerlegen lassen. Zusätzlich sind die klassischen composable Sicherheitsaussagen oft zu spezifisch und enkodieren z.B. eine spezifische angenommene oder konstruierte Ressource, was ihre Wiederverwendung erschweren kann. Einige verbreitete Abstraktionsschnittstellen, wie z.B. das Random-Oracle, sind des weiteren gänzlich unmöglich aus effektiv in der realen Welt vorhandenen Ressourcen zu konstruieren.

Letztlich wird composable Sicherheits-Frameworks teilweise auch bedingt durch die Vielzahl Unmöglichkeitsresultate mit Skepsis begegnet. So sind Ziele wie Commitment-Verfahlren und Zero-Knowledge-Beweise, welche in einer isolierten Betrachtung erreichbar sind, in composable Frameworks ohne zusätzliche Annahmen unmöglich zu erreichen. Insbesondere tritt im Kontext der adaptiven Sicherheit das sogenannte Simulator-Commitment-Problem auf: sobald eine Partei korrumpiert wird, ist ein effizienter Simulator nicht mehr in der Lage konsistent mit seinen vorherigen Ausgaben zu sein.

Erweiterungen des Constructive Cryptography Frameworks In dieser Dissertation verallgemeinern und erweitern wir das Constructive Crytpography Framework auf drei Arten, um diese Limitierungen zu beseitigen.

Zuerst führen wir eine Erweiterung um sogenannte *globale Ereignishistorien* ein, welche flexiblere Abstraktionsschnittstellen ermöglichen. Dies bedeutet, dass jede Ressource zusätzlich zum traditionellen Input-Output-Verhalten auch Ereignisse aus einer bestimmten Vordefinierter Menge auslösen kann. Das Verhalten anderer Ressourcen kann dann von dieser globalen Ereignishistorie abhängen, d.h. davon welche Ereignisse und in welcher Reihenfolge sie eingetreten sind. So kann z.B. eine Schlüsselressource (ein Modul) nun das Ereignis dass der Schlüssel geleckt wurde definieren, worauf basierend ein Kommunikationskanal (ein anderes eigentlich unabhängiges Modul) seine Sicherheitsgarantien abschwächen kann.

Zweitens führen wir das Konzept der *kontextbeschränkten Konstruktion* ein. Kontextbeschränkte Konstruktionen erlauben uns zu fassen, dass gewisse Ressourcen nur in bestimmten Menge wohldefinierten Kontexte verwendet werden kann. Solch eine Definition mit explizit beschränkten Kompositionsgarantien ist insbesondere dann nützlich, wenn man es mit einer Abstraktion wie dem Random-Oracle zu tun hat, für welches keine erreichbare schwächere aber noch nützliche Alternative bekannt ist. Als eine Anwendung zeigen wir dann, dass die Universal-Composable-Extractors (UCE), welche durch Bellare et al. als eine alternative zum Random-Oracle eingeführt wurden, als ein Spezialfall von kontextbeschränkten Konstruktionen aufgefasst werden können.

Drittens greifen wir das sogenannte Simulator-Commitment-Problem auf, welches vorwiegend im Kontext von adaptiver Sicherheit auftritt. Wir führen einen neuartigen composable Sicherheitsbegriff ein, welcher die composable Sicherheitsgarantien von Protokollen, für welche bisher nur isolierte Sicherheitsdefinitionen anerkannt waren, fassen kann. Unser Sicherheitsbegriff bietet dabei sowohl eine klare Semantik wie die entsprechenden Garantien (welche in jedem Kontext gelten) zu verstehen sind, als auch eine Kompositionstheorem. Kurz zusammengefasst führen wir *intervallbasierte* Garantien ein, welche Eigenschaften formalisieren, welche zwischen zwei Ereignissen gelten. Dadurch zwingen wir den Simulator nicht, sich zwischen den einzelnen Intervallen konsistent zu

verhalten, welches die Ursache für das Commitment-Problem ist. Auf einer technischen Ebene bauen wir dafür auf dem spezifikationsbasierten Ansatz des Constructive Cryptography Frameworks auf, wobei der Sicherheitsbeweis eines Protokoll π dem Modellieren der angenommenen Spezifikation $\mathcal{R}$ und dem Beweis dass $\pi\mathcal{R}$ in einer idealen Spezifikation $\mathcal{S}$ enthalten ist entspricht. Wir formalisieren intervallbasierte Garantien als einen neuartige Typen von Spezifikationen und arbeiten die entsprechende Theorie aus.

Ein Fallbeispiel
Zu guter Letzt betrachten wir sichere Textnachrichtenprotokolle zwischen zwei Parteien und führen dazu das erste systematische und composable Sicherheitsmodell ein. Unter Ausnützung unserer Definition der globalen Ereignishistorien entwickeln wir dabei ein einheitlicher und wiederverwendbarer Typ Sicherheitsaussage für die einzelnen Unterprotokolle. Dies steht im Gegensatz zu existierenden spielbasierenden Sicherheitsabhandlungen, wo die Sicherheitsspiele der einzelnen Unterprotokolle auf die Analyse des entsprechenden übergeordneten Protokolls massgeschneidert sind, was die Wiederverwendbarkeit in anderen Protokollen und Kontexten einschränkt. Wir verwenden dabei zusätzlich unsere intervallbasierte Spezifikationen um das Commitment-Problem zu handhaben, welches bei sicheren Textnachrichtenprotokollen aufgrund der Betrachtung von Enthüllungen der Zustände der Parteien auftritt.

Contents

Chapter 1

Introduction

1.1 Motivation

The fundamental proposition of modern-day cryptography is being a science rather than an art. A cornerstone thereof is designing *provably secure* systems, rather than relying on heuristic arguments or even hoping to achieve security by obscuring its functionality. Proving a system to be secure, however, foremost requires a mathematical definition of the desired security goals and a model of the system under consideration. Furthermore, for the design and analysis of the system to be rigorous, as with any scientific discipline, modularization is key. First, modularization allows to build and understand systems that would otherwise just be to complex to discern. Second, modularization naturally allows for a separation of concerns and for the reuse of existing solutions.

Several distinct approaches towards defining security have evolved over the course of the last decades, with game-based security definitions and composable security definitions being the two most prevalent ones. While game-based security definitions are more prevalent, composable security definitions offer a set of distinct advantages, such as a tight linking to the security of the actual application and improved modularization. Nevertheless, they also face a number of obstacles, with this thesis aims at overcoming.

1.1.1 Game-Based Security Definitions

The majority of the security definitions in cryptography are so-called game-based ones. To define the security of a primitive or scheme, one thereby defines an experiment between an adversary (sometimes also called challenger) and a security game that provides a certain well defined interface for using the primitive to the adversary. The scheme is then said to be secure if no efficient (for some notion of efficiency, such as polynomial time) adversary can achieve a certain goal—called winning the game—with non-negligible probability.

Game-based notions' primary strength is their simplicity. Rather than considering a real-world usage of the primitive with a potentially complex interaction pattern, the security games strive to focus on the essential parts required to guarantee security. As a consequence, they are particularly suited to serve as a technical reference point, both for designing schemes that provably satisfy the definition (based on some assumptions), and for relying on any scheme that satisfies the definition to building more involved ones.

The main drawback of game-based definitions, however, is the disconnection of the security definition from the actual use-case. That is, while the design of a security game is commonly motivated by an application story, their connection is typically only informally argued as part of the exposition illustrating and justifying the definition. Nevertheless, it is frequently assumed that a scheme satisfying the corresponding definition then implies security in the intended application, despite the apparent lack of scientific evidence thereof. Moreover, frequently several related security definitions emerge, posing the question which one is the right one for the application in mind. While the technical relations among those definitions are usually studied, this does not settle whether in an intended application the stronger definition is required or the weaker one would suffice as well. For instance, if we can assume authenticated communication and want to communicate confidential information, it is not apparent whether IND-CCA or IND-CPA security represents the correct reference point to aim for.

In addition, game-based security definitions do not compose by default. Consider the example of designing a scheme for secure communication, i.e., one that provides both authenticity and confidentiality, in a modular manner. For this, one needs a security definition for

the authentication scheme, one for the encryption scheme, as well as one for the overall scheme. It then needs to be explicitly proven—via reductions—that breaking the overall scheme implies breaking one of the underlying scheme. If glossed over, the scheme might even be insecure despite each of its component being provably secure.

1.1.2 Composable Security Definitions

Composable security frameworks bridge the aforementioned gap between the technical security definitions and the actual use of a scheme by taking a radically different—yet completely natural—approach to defining security. Several such frameworks exists, such as the Universal Composability (UC) framework by Canetti and its variants thereof [Can01; CR03; CDPW07; HS15], the Reactive Simulatability (RSIM) framework by Backes, Pfitzmann, and Waidner [PW01; BPW07], the Constructive Cryptography (CC) framework by Maurer and Renner [MR11; Mau11; MR16], and the IITM model by Küsters, Tuengerthal, and Rausch [KTR13; CEK+16; CKKR19].

While these frameworks have been developed with different focuses in mind and diverge on a technical level, on a high level they all follow the real-world ideal-world paradigm for defining security. To this end, they model an actual execution of the protocol in the real world. In that model, the protocol makes use of certain *assumed resources*, such as different types of communication channels and shared keys. This real-world execution is then compared to an execution of an idealized resource, asserting that they behave equally in any environment. Hence, the protocol can be viewed as constructing the idealized resource, thus often called *constructed resource*, from the assumed ones. Such a constructive perspective puts cryptography on par with other engineering disciplines, where posing the questions of which resources are needed to obtain the desired product is commonplace.

The foremost advantage of composable security definitions is their evident semantics. First, as composable security definitions consider all environments, it is ensured that the constructed resource can truly be used. For instance, if a key-exchange protocol constructs a key resource, then this key can be safely used in any arbitrary application, such as the one-time pad, which is surprisingly not always true when working with game-based definitions. Second, they make the assumed resources

explicit as part of the security statement, rather than encoding them implicitly as part of a security game. For instance, if a composable statement about an encryption scheme assumes authenticated channels, then this requirement on the scheme's suitability is apparent and clearly needs to be addressed in any usage. In contrast, in a game-based world the imposed restrictions from a scheme only satisfying IND-CPA rather than a stronger notion, such as IND-CCA, tend to be much less obvious. Third, the constructed resource of a composable security statement explicitly specifies the desired behavior and achieved security properties, rather than listing excluded attacks as game-based security definitions tend to do. For instance, in context of encryption the constructed resource is typically a secure channel that specifies that the receiver gets the correct message if the ciphertext is delivered (correctness), that the adversary learns at most the length of the messages (confidentiality), and the the adversary can at most drop, deliver, or reorder messages (authenticity). Given such a constructed resource, it is self-evident that the message length might leak.

A second key advantage of composable security frameworks is, that they inherently foster modularity and abstraction. Both of those aspects are widely recognized to be crucial for designing and analyzing complex systems—across all engineering disciplines well beyond cryptography. Composable frameworks are based around defining components with clean abstraction boundaries (e.g., a secure channel) that abstract away the details of how that module has been constructed or otherwise obtained. This idealized module can then be used by a higher-level protocol with the security of the combined overall protocol following directly from the composition theorem. Given that composable frameworks enforce a uniform type of statement, that moreover not only makes the analysis of a complex protocol modular, but also enables the sub-protocols to be reused as building blocks for other protocols. This directly calls for a library of such construction statements as a main objective for cryptography.

Constructive Cryptography

In this thesis, we build on the Constructive Cryptography (CC) framework by Maurer and Renner [MR11; Mau11]. Compared to other composable frameworks, it provides two major advantages.

First, it is built around the paradigm of top-down abstraction, in contrast to other frameworks that are defined bottom-up from a concrete computational model. Using top-down abstraction allows treating each statement at the right level of abstraction. As irrelevant aspects are abstracted away, this leads to cleaner descriptions and minimal proofs that avoid unnecessary technicalities. In this thesis, we mainly work on the abstraction layer of the input-output behavior of concrete systems. In the corresponding abstraction level of CC, this behavior is the mathematical object under consideration, rather than a property of a concrete computational object. In particular, this implies that our statement are valid independent of the actual implementation of the systems. Moreover, the abstract approach of CC does not hard-code an efficiency notion—forgoing it for explicit statements entailing reductions. Hence, it avoids being restricted to asymptotic statements that do not reflect real-world use of the protocol with concrete parameters.

Second, Constructive Cryptography brings the natural idea of specifications to the realm of cryptography. Considering specifications, i.e., sets of objects, and abstracting them by larger sets that are easier to understand is a widely used approach to make composable statements outside cryptography. In particular, in such an approach composition is just the transitivity of the subset relation: if one set $\mathcal{R}$ can be abstracted by $\mathcal{S}$, i.e., $\mathcal{R} \subseteq \mathcal{S}$, and as another statement one shows $\mathcal{S} \subseteq \mathcal{T}$, then $\mathcal{R} \subseteq \mathcal{T}$ is directly guaranteed. This stands in stark contrast to the traditional, and rather technical, composition theorems of most other composable frameworks.

Obstacles for their Adaption

The way composable security frameworks provide readily understandable security guarantees and foster modularity, might make them look like the only viable approach to a rigorous and sound design of cryptographic applications. Nevertheless, they have not seen wide-spread adaption and are often even met with some skepticism.

In general, the tightly restricted type of statements they enforce have proven to be both a blessing and a curse. First, the imposed abstraction boundaries lack flexibility. This impedes modularity, since it prevents certain resources from being properly decomposed. Moreover, some

popular abstraction boundaries, such as the random oracle, are known to be impossible to construct from resources readily available in the real world; in particular, instantiating them with hash functions is provably unsound. Nevertheless, the random oracle paradigm seemingly leads to secure protocols in practice, indicating that the random oracle resource is not the correct abstraction boundary. This raises the question about more amenable abstraction boundaries that preserve (most of) the benefits of composable security frameworks.

Second, reusability is often hindered by enforcing overly specific statements. In particular, a traditional composable statement is generally phrased as the resource constructed when assuming that the protocols makes use of a particular resource. This, however, implies that whenever the protocol is used in a setting where an even slightly different resource is available, than the one from the security proof, then the statement no longer applies. This raises the question how one can phrase statements that do not unnecessarily hard-code a fixed assumed resource, but rather describe what the protocol achieves when run in an arbitrary context, without greatly complicating the analysis.

Third, composable security definitions are met with some skepticism because of many impossibility results; goals such as commitments and zero-knowledge that are achievable in a stand-alone sense were shown to be unachievable composably (without a setup) since provably no efficient simulator exists. In particular, in the context of adaptive security, the so-called "simulator commitment problem" arises: once a party gets corrupted, an efficient simulator is unable to be consistent with its pre-corruption outputs. A natural question is whether such impossibility results are unavoidable or only artifacts of frameworks being overly restrictive.

1.2 Overview and Contributions

This thesis makes both conceptual and technical contributions to the definitional aspects of cryptography. This includes several extensions to the Constructive Cryptography framework to address the aforementioned obstacles of composable security definitions. In the following, we summarize the main contributions and give pointers to the relevant papers they are based on.

1.2.1 Advancing Composable Security

Constructive Cryptography with Events

In Chapter 3, we propose the extension of the Constructive Cryptography framework by so-called global event histories, to enable more flexible abstraction boundaries. In particular, we tackle the problem that in many cryptographic systems or protocols the behaviors of components are subtly intertwined. For example, a channel (one module) can become insecure when a key (another, apparently independent module) is leaked to the adversary. In the standard system model of Constructive Cryptography (and most other composable frameworks), all resources are however by definition fully independent. The only way for two resources to interact is via explicit communication through the protocol or the environment. While having such a clear and well-defined abstraction boundary is crucial for modularity, the traditional system model implies that seemingly natural modules (e.g., the channel and the key from the previous example) cannot be modeled as separate components, but have to be modeled as a monolithic resource. In summary, the existing system model frequently impedes modularization, and thus reuse, by being to coarse grained.

As a solution, we propose to enhance the abstraction boundary by global events. That is, in addition to the traditional input-output behavior, each resource can trigger events from a predefined set. In the new system model we propose, those events are globally observable. Namely, other resources can then depend on the global event history, i.e., on which events having occurred so far and in which order. For instance, the key resource can then export the event of having been leaked, on which the behavior of the channel can then depend on, without this information having to be somehow routed via some wrapper. Global events, therefore, allow extending the abstraction boundary in a fine-grained fashion, with the designer of the resource choosing which events are exported as part of the abstraction interface, greatly enhancing the flexibility of the resource paradigm. Since Constructive Cryptography is build around top-down abstraction, this novel system implementation furthermore inherits all the properties proven on the more abstract layers.

The results in that chapter are based on the publication [JMM19b].

Context-Restricted Constructions

In Chapter 4, we propose the notion of context-restricted constructions. Context-restricted constructions allow us to model that certain resources do not compose generally, i.e., cannot be used in any arbitrary context. To this end, context-restricted constructions treat the admissible contexts as an explicit parameter of the security statement, making the remaining composition guarantees explicit. A context thereby restricts the honest parties' use of the resource, i.e., specifies a set of protocols for which it is safe to use the resource. It, however, never restricts the adversary and, hence, preserves the clean semantics of composable security statements. In summary, a context-restricted construction guarantees that when used by the honest party in one of the explicitly specified manners, the protocol does construct the ideal resource with all its guarantees, even in the presence of an arbitrary adversary.

Such a notion with explicitly limited composition guarantees is particularly useful when dealing with abstractions such as a random oracle, where no weaker and feasible to instantiate, yet useful abstraction boundary are known. As an application, we then show that the notion of universal computational extractors (UCE), introduced by Bellare et al. [BHK13b] as an alternative to the ROM, can be understood as a special case of context-restricted constructions about random oracles. UCE is multi-stage game-based notion, aimed at formalizing classes of applications for which the ROM paradigm might be sound. To this end, UCE is a parametrized notion, where each so-called UCE family specifies a set of admissible split-adversaries. We show a generic transformation from UCE families to context restrictions, thereby giving the multi-stage UCE game clear composable semantics. Furthermore, taking the viewpoint of general context-restricted constructions directly leads to consider natural context restrictions that cannot be phrased within the traditional UCE game. Finally, we prove a technical result about the Merkle-Damgård construction. Namely, we show that if the compression function satisfies one of our generalizations of UCE (for which no impossibility result is known), then the overall construction satisfies one of the of the well-known UCE variants. Thus, using our novel notion we further validate the Merkle-Damgård construction, being able to base the overall security on a plausible assumption.

The results in that chapter are based on the publication [JM18].

Overcoming the Commitment Problem

In Chapter 5 we revisit the so-called simulator commitment problem, that mainly occurs in the context of adaptive security. Since the commitment problem usually occurs at a very specific point of the protocol execution, such as when a party gets corrupted, we propose a novel construction notion centered around the very natural idea of formalizing guarantees that hold in a certain interval (between two events), yet compose. To this end, we leverage that the Constructive Cryptography framework's specification-based approach, where proving a protocol π to be secure corresponds to modeling the assumed real-world specification $\mathcal{R}$, and showing that the resulting specification $\pi\mathcal{R}$ is contained in an ideal-world specification $\mathcal{S}$. While most composable frameworks only consider a single type of specifications based round the existence of a simulator, we use the additional flexibility of CC to consider specifications formalizing interval-wise guarantees. We develop the required theory, carefully considering the subtleties arising when defining such specifications, and present the corresponding composition theorem. Moreover, we discuss our notion's security guarantees should be interpreted, when stronger notions might still be desirable, and how our notion fits into the space of static versus adaptive security.

We then apply our methodology to several examples. First, we consider the encrypt-then-MAC paradigm in a setting where the keys can adaptively leak to the adversary, stylizing adaptive passive corruptions. Using our interval-wise guarantees, we obtain a simple composable security definition thereof without the need for non-committing encryption. As a second application, we present a composable formalization of information-theoretically binding commitment schemes realizable in the plain model. We then show how, based on such a commitment scheme, Blum's protocol [Blu83] constructs a composable coin-toss notion. Finally, we consider the composable guarantees of identity-based encryption. We revisit the result by Hofheinz, Matt, and Maurer [HMM15] that shows the standard ind-id-cpa notion to be too weak when considering a traditional composable statement, even when considering static corruptions, due to the commitment problem. Based on interval-wise guarantees we formalize a composable specification of IBE that corresponds exactly to the standard ind-id-cpa notion.

A preprint of the results in that chapter is available in [JM20].

1.2.2 A Case Study: Secure Messaging

In Chapter 6, we study security definitions for secure messaging. Secure-messaging protocols aim at providing strong security guarantees to two parties that asynchronously communicate over an insecure network. Apart from protecting the messages' confidentiality and integrity, the desired properties include forward secrecy and healing from a state or randomness exposure. The latter properties are addressed by having the parties continuously update their keying material.

Secure messaging has attracted a fair bit of attention from the cryptographic community, initiated by the first formal analysis of the widely-used Signal protocol [OWS17] by Cohn-Gordon et al. [CCD+17]. Influence by the Signal protocol, a number of other secure messaging protocols with stronger guarantees [Poe18; JS18; DV18; JMM19a] have been proposed. Furthermore, a line of work considers continuous key-agreement as an abstraction of the Signal protocol, proposing and analyzing various protocols [BSJ+17; Poe18; ACD19] with different levels of security. The majority of the proposed protocols pursue similar goals, but each achieves a slightly different trade-off between security, efficiency and usability. Moreover, each construction comes with its own—fairly complex—security game, intermediate abstractions, and sub-protocols. In particular, the security game of each sub-protocol is typically co-designed with the overall protocol and implicitly encodes the security guarantees provided by other sub-protocols. This not only renders them hard to compare, but additionally prevents achieving new trade-offs that would result from combining sub-protocols from different works without reproving their security.

In this thesis, we thus pursue to initiate the build of a library of composable and reusable statements in lieu of tailor-made security games. First, we demonstrate the applicability of global events to this task. Secure messaging protocols typically have a plenty of intra-module dependencies, as for instance, the confidentiality of a message typically depends on whether and when a party's state leaked and other messages being sent and delivered. Constructive Cryptography with global events, thus, aids modularization, as it allows us to model the relevant dependencies as events, while abstracting other implementation details. The notion of global events also provide a solution to the problem of overly specific statements in the context of secure messaging. To this end,

we parameterize the resources by several (discrete) parameters, which can be thought of as a switch with two or more positions, that depend on the global event history. Each such switch thereby downgrades the security of a resource, e.g. switch a channel from non-leakable (i.e. confidential) to leakable. The goal of (sub-)protocol is then phrased as improving a certain parameter while leaving the other parameters unchanged, independently of what they are, avoiding hard-coding of any unnecessary assumptions.

Finally, as one considers adaptive state exposure, secure messaging is prone to the simulator commitment problem. We thus formalize several of our statements using interval-wise guarantees. In addition, we also introduce a novel protocol that avoids the commitment problem, and thus satisfies an even stronger specification, using non-committing In addition, we propose a technique that allows to transform many standard SM protocols into protocols that avoid the commitment problem and, thus, satisfies an even stronger specification. This however comes at the expense of an efficiency lost, as well as being restricted to only sending a bounded number of messages before receiving a reply from the other party, further demonstrating the utility of being able to making weaker statements using interval-wise guarantees.

The results in that chapter are based on the publication [JMM19b].

1.3 Related Work

A number of approaches have been proposed in order to overcome various limitations of composable security definitions. This thesis is divided in chapters that tackle different shortcomings of the standard composable security frameworks. As a consequence, we provide the relevant related work in the respective chapters. The bibliography can be found at the end of the thesis.

Chapter 2

Preliminaries

2.1 Notation

2.1.1 General Notation and Information Theory

We denote the set of natural numbers by $\mathbb{N} = \{1, 2, \ldots\}$, and the set of integers by $\mathbb{Z}$. For $n \in \mathbb{N}$, we use the convention $[n] \coloneqq \{1, ..., n\}$. For a family of sets $\mathcal{X}_i$, $i \in \mathcal{I}$, we denote by $\biguplus_{i \in \mathcal{I}} \mathcal{X}_i$ the disjoint union. Finally, $\langle X_i \rangle_{i \in \mathcal{I}}$ denotes the tuple that assigns X_i to the index $i \in \mathcal{I}$

The probability of an event A in an experiment E is denoted by $\Pr^E[A]$. If the experiment is clear from the context, we omit the superscript. For a discrete random variable X distributed over a carrier set $\mathcal{X}$, we denote by P_X the probability mass function of X, i.e., $\mathsf{P}_X(x) \coloneqq \Pr[X = x]$. The conditional probability of A given B is denoted $\Pr[A \mid B]$ and for discrete random variables X and Z, the conditional probability distribution of X given Z is the partial function $\mathsf{P}_{X|Y}(x, y) \coloneqq \Pr[X = x \mid Y = y]$.

The expected value of a random variable X is denoted by $\mathbb{E}[X]$. We denote the entropy of a discrete random variable X distributed over $\mathcal{X}$ as $H(X) \coloneqq -\sum_{x \in \mathcal{X}} \mathsf{P}_X(x) \log_2(\mathsf{P}_X(x))$. Moreover, we denote by $H_\infty(X) \coloneqq -\log_2(\max_{x \in \mathcal{X}} \mathsf{P}_X(x))$ the min-entropy of X. Finally, we denote by $\tilde{H}_\infty(X)$ the average min-entropy of X given Z, defined as $\tilde{H}_\infty(X \mid Z) \coloneqq -\log_2(\mathbb{E}_{z \in \mathcal{Z}}[\max_{x \in \mathcal{X}} \mathsf{P}_{X|Z}(x, z)])$.

2.1.2 Pseudo-Code for Algorithms and Systems

We describe algorithms and systems using pseudo-code. The following conventions are followed: We write $x \leftarrow y$ for assigning the value y to the variable x. For a finite set $\mathcal{X}$, $x \twoheadleftarrow \mathcal{X}$ denotes assigning x uniformly at random a value in $\mathcal{X}$. Furthermore, $x \overset{\mathsf{P}_X}{\twoheadleftarrow} \mathcal{X}$ denotes sampling x according to the indicated probability distribution P_X over $\mathcal{X}$.

To describe reactive discrete systems, such as resources, we adhere to the following conventions: Each system has one or several interfaces to interact with, and an initialization procedure initializing all the persistent variables (all other variables are understood to be volatile). Formally this initialization is called upon querying the system at any interface for the first time. Typically queries, also called inputs, to an interface of a systems consist of a suggestive keyword and a list of arguments, such as for a channel the input $(\texttt{send}, m)$ to send a message m. We ignore keywords in writing the domains of arguments, e.g., $(\texttt{send}, m) \in \mathcal{M}$ indicates that $m \in \mathcal{M}$. Furthermore, we use the **require** command, which should be understood as a shortcut for explicitly tracking the respective condition and returning an error symbol $\perp$ in case the condition is violated. Finally, we use the syntax "**call** $y \leftarrow (\texttt{kwd}, x)$ at interface I of R" to denote that the system queries $(\texttt{kwd}, x)$ at the interface I of another system R and stores the returned value in the variable y.

See also Section 2.2.6 for an additional introduction of some of the conventions used, by the means of an example.

2.2 Constructive Cryptography

This thesis is centered around composable security notions. There are several frameworks formalizing composable security, sometimes also called simulation-based security, such as [Can01; PW01; BPW07; MR11]. While they share many common aspects, and most of the ideas proposed in this thesis could potentially be translated among the frameworks, our presentation is focused here solely on the Constructive Cryptography framework [MR11; Mau11; MR16]. The Constructive Cryptography framework is built around constructions of resource specifications (from other resource specifications). A specification is thereby simply a set of

objects called resources. Each resource, such as communication channels or computational resources, is accessible to one or more parties, and presents a mathematical abstraction of an object either found in the physical world or obtained via the means of a construction.

The Constructive Cryptography framework is a general theory that expands beyond cryptography and consists of several abstraction layers in a top-down fashion. The purpose of this section is to describe the aspects of Constructive Cryptography relevant for this work, thereby mainly staying at the abstraction level of probabilistic discrete systems and focusing on its applicability on cryptography. We refer to [MR11; MR16] for a detailed presentation of the higher axiomatic abstraction layers and its reach beyond cryptography.

2.2.1 Specifications.

A general paradigm in science and engineering is the concept of a specification. A specification thereby captures certain assumed or desired properties of a system. Following [MR16], we model specifications as sets of objects $\mathcal{R} \subseteq \Theta$, where Θ denotes some basic set of objects under consideration. For each property one has in mind, one can consider the set $\mathcal{R}$ of objects satisfying that property. Vice versa, each set of objects $\mathcal{R}$ can be interpreted as the set of properties common to all elements. A concrete object X can then either satisfy the specification, i.e., $\mathsf{X} \in \mathcal{R}$, or not. For instance, one might consider the set of all programs and the specification of all programs $\mathcal{R}$ exhibiting a certain input-output behavior. Functional correctness of a program P then corresponds to $\mathsf{P} \in \mathcal{R}$. In cryptography, one might consider the set of all communication channels and model authenticity as the set of all communication channels that guarantee the correct messages (or no message at all) to be delivered.

An fundamental concept is then principle of *abstracting* a specification $\mathcal{R}$ by another one $\mathcal{S}$, i.e., $\mathcal{R} \subseteq \mathcal{S}$. In other words, $\mathcal{S}$ waives some of the (irrelevant) guarantees of $\mathcal{R}$ in favor of a less complex and more useful one. For instance, $\mathcal{R}$ might be a specification of communication channels that precisely models the capabilities of the various parties, while $\mathcal{S}$ might specify authenticity only, ignoring other aspects not relevant for the further analysis.

2.2.2 The System Algebra

At its heart, the Constructive Cryptography framework views cryptography as a resource theory with each specification being of a set of resources. Resources are part of a system algebra consisting of resources and converters that can be combined to form new systems, adhering certain natural algebraic rules. In this thesis, we focus on the abstraction layer of discrete probabilistic systems, defined by their input-output behavior.

Resources

The central object in Constructive Cryptography is that of a resource. A resource R is a reactive discrete system that has one or several *interfaces*, each assigned to a party. For instance, a communication channel might have a sender's interface at which a message can be input, a receiver's interface at which it can be read afterwards, and an adversarial interface at which some leakage might be obtained. More generally, for each party $P \in \mathcal{P}$, the resource R has a set of interfaces $\mathcal{I}_P$ via which it interacts with the environment in the following way: The environment can repeatedly input queries x at one of the interfaces, to obtain the corresponding response y at the same interface. This reply y can thereby (probabilistically) depend on all previous inputs and outputs. Formally, a resource for parties $\mathcal{P}$ with interface sets $\langle \mathcal{I}_P \rangle_{P \in \mathcal{P}}$ and input and output alphabet $\mathcal{X}$ is a special type of a random system [Mau02], where the interface address is encoded as part of the inputs.

Definition 2.2.1. Let $\mathcal{X}$ denote a countable alphabet, let $\mathcal{P}$ denote a finite set of parties, and let $\langle \mathcal{I}_P \rangle_{P \in \mathcal{P}}$ denote a tuple of finite interface sets. Moreover, let $\mathcal{I} \coloneqq \bigcup\!\!\bullet_{P \in \mathcal{P}} \mathcal{I}_P$. A probabilistic $(\mathcal{X}, \mathcal{P}, \langle \mathcal{I}_P \rangle_{P \in \mathcal{P}})$-resource S is a sequence of partial conditional probability distributions

$$\left\langle \mathsf{P}^{\mathsf{S}}_{Y_i | X^i Y^{i-1}} : \left(\mathcal{X} \times (\mathcal{I} \times \mathcal{X})^i \times \mathcal{X}^{i-1} \right) \to [0,1] \right\rangle_{i \geq 1},$$

such that $\mathsf{P}^{\mathsf{S}}_{Y_i | X^i Y^{i-1}}(y_i; x^i, y^{i-1})$ is defined if and only if

$$\prod_{j=1}^{i-1} \mathsf{P}^{\mathsf{S}}_{Y_j | X^j Y^{j-1}}(y_j; x^j, y^{j-1}) > 0,$$

i.e., when not conditioning on an impossible history.

A finite set of resources with disjoint interface sets can be viewed as a single one that provides each party access to the corresponding interfaces of all subsystems. That is, consider $(\mathcal{X}, \mathcal{P}, \langle \mathcal{I}_{P,i} \rangle_{P \in \mathcal{P}})$-resources R_i, for $i \in [n]$, such that $\mathcal{I}_{P,i} \cap \mathcal{I}_{P,j} = \emptyset$, for all $j \neq i$ and $P \in \mathcal{P}$. Then, we denote by $[\mathsf{R}_1, \ldots, \mathsf{R}_n]$ the *parallel composition*, which is a $\big(\mathcal{X}, \mathcal{P}, \langle \bigcup_{i \in [n]} \mathcal{I}_{P,i} \rangle_{P \in \mathcal{P}}\big)$-resource.

We furthermore denote by $\square$ the (canonical) dummy resource, where every party has an empty interface set. By definition, we thus have that $[\mathsf{R}, \square] = \mathsf{R}$ for all resources R.

Converters and Protocols

A party can use a resource R by applying a *converter* at its interfaces. Such a converter can be thought of a protocol engine. The guiding principle of Constructive Cryptography is, thereby, that everything is considered to be relevant for an analysis is modeled as a resource. In contrast, everything modeled as part of the converter is defined to be intentionally ignored for the sake of the analysis. For instance, in cryptography one often chooses to neglect the exact computational cost of a certain operation as long as it is "efficient", where efficiency is then traditionally defined as the asymptotic notion (in some security parameter) of polynomial time. In this work, we mainly avoid the delicate task of choosing the class of converters under consideration by making all converters explicit.

On an abstract level, attaching a converter to a set of interfaces of a resource yields a new resource, i.e., converts the resource. As with resources, we focus on the more concrete layer of discrete systems here and define converters accordingly. A converter π is modeled as a probabilistic discrete system with two sets of interfaces: the inner interfaces $\mathcal{I}_{\mathsf{in}}$ and the outer interfaces $\mathcal{I}_{\mathsf{out}}$. Upon an input at an outside interface, the converter is allowed to make a bounded number of queries to the inside interfaces (recall that a resource always returns at the same interface it was queried), before returning a value at the queried interface.

Definition 2.2.2. A $(\mathcal{X}, \mathcal{I}_{\mathsf{in}}, \mathcal{I}_{\mathsf{out}})$-converter is a system whose input alphabet at the beginning is $\mathcal{X} \times \mathcal{I}_{\mathsf{out}}$. Upon an input (x, I), the converter produces an output, either from $(y, I') \in \mathcal{X} \times \mathcal{I}_{\mathsf{in}}$ (interpreted as input y

for the corresponding interface of the system connected at the inside) or of the from $y \in \mathcal{X}$ (interpreted to be the response at I). After an output of the form (y, I), the converter takes an input $x' \in \mathcal{X}$ (interpreted as the response from the connected resource), and after an output of the form y, the converter takes another input from $\mathcal{X} \times \mathcal{I}_{\mathsf{out}}$, i.e., from the outside. There is a finite upper bound on the number of consecutive outputs of the form $(y, I') \in \mathcal{X} \times \mathcal{I}_{\mathsf{in}}$.

Converters can be attached to a resource in order to modify the behavior of the resource. More specifically, for a $(\mathcal{X}, \mathcal{P}, \langle \mathcal{I}_P \rangle_{P \in \mathcal{P}})$-resource R and a $(\mathcal{X}, \mathcal{I}_{\mathsf{in}}, \mathcal{I}_{\mathsf{out}})$-converter π, let $\gamma_P \colon \mathcal{I}_{\mathsf{in}} \hookrightarrow \mathcal{I}_P$ be an injective function, such that $(\mathcal{I}_P \setminus \mathrm{img}(\gamma)) \cap \mathcal{I}_{\mathsf{out}} = \emptyset$, describing the connections between the converter's inside interfaces and a subset of the resource's interfaces. Then, $\mathsf{R}' \coloneqq \pi^{\gamma}\mathsf{R}$ denotes the $(\mathcal{X}, \mathcal{P}, \langle \mathcal{I}_P \rangle'_{P \in \mathcal{P}})$-resource, where $\mathcal{I}'_P = \mathcal{I}_P \setminus \mathrm{img}(\gamma) \cup \mathcal{I}_{\mathsf{out}}$ and $\mathcal{I}'_Q = \mathcal{I}_Q$ for $Q \neq P$, obtained from connecting the converter accordingly.

Converter attachment satisfies the natural property of *composition order independence*, stating that on the term algebra level the composition order does not matter—only the final system. This is summarized by the following proposition.

Proposition 2.2.3. *Let P and Q be two different parties, let π_P and π_Q be two converters, and let R be a resource. Then for any suitable γ_P and γ_Q we have*

$$\pi_P^{\gamma_P} \pi_Q^{\gamma_Q} \mathsf{R} = \pi_Q^{\gamma_Q} \pi_P^{\gamma_P} \mathsf{R}.$$

Moreover, if S is another resource such that the interface sets of R and S are disjoint, then we have

$$\pi_P^{\gamma_P} \left[\mathsf{R}, \mathsf{S}\right] = \left[\pi_P^{\gamma_P} \mathsf{R}, \mathsf{S}\right].$$

We define a *protocol* to be a (partial) tuple of converter-connection pairs. That is, for a subset of parties $\mathcal{Q} \subseteq \mathcal{P}$, let $\boldsymbol{\pi} \coloneqq \langle (\pi_P, \gamma_P) \rangle_{P \in Q}$, and write $\boldsymbol{\pi}\mathsf{R} \coloneqq \pi_{P_1}^{\gamma_{P_1}} \cdots \pi_{P_n}^{\gamma_{P_n}} \mathsf{R}$, where $n = |\mathcal{Q}|$, to denote its application. (By composition order independence it is irrelevant in which order the converters are attached.) For two protocols $\boldsymbol{\pi}$ and $\boldsymbol{\pi'}$, we denote by $\boldsymbol{\pi'} \circ \boldsymbol{\pi}$ their sequential composition, i.e., the protocol for which $(\boldsymbol{\pi'} \circ \boldsymbol{\pi})\mathsf{R} = \boldsymbol{\pi'}(\boldsymbol{\pi}\mathsf{R})$ for any resource R. Moreover, we denote by $\boldsymbol{id}$ the identity protocol, for which $\boldsymbol{id}\mathsf{R} = \mathsf{R}$, for any resource R.

2.2.3 The Construction Notion

While the concept of abstracting specifications $\mathcal{R}$ by another one $\mathcal{S}$ is a fundamental principle of many scientific fields, in cryptography one is usually interested in the more concrete aspects of constructions or constructability. That is, given some assumed specification $\mathcal{R}$, is there a protocol π that constructs $\mathcal{S}$ from $\mathcal{R}$, in short $\pi\mathcal{R} \subseteq \mathcal{S}$? Hence, the Constructive Cryptography framework introduces the following construction notion as a shorthand notation.

Definition 2.2.4. Let $\mathcal{R}$ and $\mathcal{S}$ be arbitrary specifications, and let π be an arbitrary protocol for $\mathcal{R}$. Then, we say that π *constructs* $\mathcal{S}$ from $\mathcal{R}$, denoted $\mathcal{R} \overset{\pi}{\longmapsto} \mathcal{S}$, if and only if $\pi\mathcal{R} \subseteq \mathcal{S}$, i.e.,

$$\mathcal{R} \overset{\pi}{\longmapsto} \mathcal{S} \;:\Longleftrightarrow\; \pi\mathcal{R} \subseteq \mathcal{S}.$$

This construction notion is associated with the usual composition properties of Constructive Cryptography: sequential and parallel composition—which form the equivalence of the universal composition theorem of the UC-framework.

Theorem 2.2.5. *Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be arbitrary specifications, and let π and π' be arbitrary protocols for $\mathcal{R}$ and $\mathcal{S}$, respectively. Then, we have*

1. $\mathcal{R} \overset{\pi}{\longmapsto} \mathcal{S} \wedge \mathcal{S} \overset{\pi'}{\longmapsto} \mathcal{T} \implies \mathcal{R} \overset{\pi' \circ \pi}{\longmapsto} \mathcal{T}$,
2. $\mathcal{R} \overset{\pi}{\longmapsto} \mathcal{S} \implies \big[\mathcal{R}, \mathcal{T}\big] \overset{\pi}{\longmapsto} \big[\mathcal{S}, \mathcal{T}\big]$.

Proof. The first property follows directly from the transitivity of the subset relation, $\pi'\big(\pi\mathcal{R}\big) \subseteq \pi'\mathcal{S} \subseteq \mathcal{T}$, and the second property follows from Proposition 2.2.3: $\pi\big[\mathcal{R}, \mathcal{T}\big] = \big[\pi\mathcal{R}, \mathcal{T}\big] \subseteq \big[\mathcal{S}, \mathcal{T}\big]$. □

The construction notion, furthermore, can be seen as a generalization of the so-called real-world / ideal-world paradigm used in most composable security frameworks [Can01; PW01; MR11; KTR13]. To this end, $\pi\mathcal{R}$ can be understood as a mathematical model of a real-world execution of a protocol, that use some assumed resources, while $\mathcal{S}$ can be seen as an ideal world giving the desired security properties by definition. Following the paradigm, the construction statement then

affirm that the real word is "just-as-good" as the ideal world, meaning that for all parties, no matter whether honest or adversarial, it does not make a difference whether they live in the real (where an arbitrary element of $\pi\mathcal{R}$ is present), or in the ideal world (where some element of $\mathcal{S}$ is present). Hence, if the honest parties are content with the guarantees they get from the ideal specification, they can safely execute the protocol in the real world instead.

2.2.4 Relaxations

While many composable frameworks hard-code a particular computational model and security flavor—e.g., computational security with negligible distinguishing advantage for all polynomial time environments—the construction notion of CC, on the other hand, is absolute. It does not require the choice of a particular computational model, an (asymptotic) efficiency notion, a particular type of indistinguishability (computational versus statistical), or even the existence of a simulator. Those aspects are formalized by so-called relaxations, as introduced in [MR16]. Treating relaxations as orthogonal to the basic construction notion allows the framework to consider different security notions within the same overall framework and, more generally, enables more flexible security statements.

On an abstract level, a relaxation is a mapping from specifications to weaker, so-called relaxed, specifications. For our purpose, where we instantiate specifications by sets of resources, we can define a relaxation as a function mapping a single resource to a set of resources.

Definition 2.2.6. Let Θ denote the set of all resources. A *relaxation* ϕ is a function $\phi\colon \Theta \to 2^{\Theta}$ (where 2^{Θ} denotes the power set of Θ) such that $\mathsf{R} \in \phi(\mathsf{R})$ for all $\mathsf{R} \in \Theta$. In addition, for a specification $\mathcal{R}$, we define $\mathcal{R}^{\phi} \coloneqq \bigcup_{\mathsf{R}\in\mathcal{R}} \phi(\mathsf{R})$ as a shorthand notation.

A concrete relaxation thereby formalizes some notion of resources being "almost-as-good" in some context. That is, if one was happy with constructing a resource specification $\mathcal{S}$, then one should also be happy with $\mathcal{S}^{\phi}$, if ϕ is believed to be justifiable in the given context. For instance, one could consider the relaxation that maps the resource R to the set of all computationally indistinguishable resources from R under some computational assumption. Then, if we consider the

ideal specification modeling a secure channel, its relaxation consists of the set of all systems indistinguishable from a secure channel. Thus, showing that the real-world system is contained therein, asserts that it behaves like a secure channel unless the computational assumption is broken. Hence, if one believes the computational assumption to be valid, one should be as content with the relaxed specification as with secure channel.

In the spirit of abstracting away irrelevant properties as a core paradigm of any modular analysis, relaxations can be "ignored" in a sequence of construction steps. That is, if one shows that one protocol constructs $\mathcal{S}^{\phi}$ (from some assumed resources), one can compose it with another statements that assumes $\mathcal{S}$ instead, and later reapply the relaxation. On the most abstract level, this follows from the following rules applying to any relaxation.

Proposition 2.2.7. *For any specifications $\mathcal{R}$ and $\mathcal{S}$, and any relaxation ϕ, we have*

1. $\mathcal{R} \subseteq \mathcal{R}^{\phi}$,
2. $\mathcal{R} \subseteq \mathcal{S} \Longrightarrow \mathcal{R}^{\phi} \subseteq \mathcal{S}^{\phi}$,
3. $(\mathcal{R} \cap \mathcal{S})^{\phi} \subseteq \mathcal{R}^{\phi} \cap \mathcal{S}^{\phi}$,
4. $(\mathcal{R} \cup \mathcal{S})^{\phi} = \mathcal{R}^{\phi} \cup \mathcal{S}^{\phi}$.

Proof. All properties follow directly from basic set theory and the fact that by definition $\mathsf{R} \in \phi(\mathsf{R})$. □

The reduction relaxation.

The most important relaxation for this thesis captures computational security based on explicit reductions. To this end, we consider *distinguishers*. A distinguisher D for resources with interfaces $\mathcal{I}$ is a system that can be attached to them with one additional interface, where it outputs a single bit after interacting with the resource.

Definition 2.2.8. The distinguishing advantage of a distinguisher D for resources R and S (both of them having the same interface set) is defined as

$$\Delta^{\mathsf{D}}\big(\mathsf{R},\ \mathsf{S}\big) \coloneqq \Pr\big[\mathsf{D}(\mathsf{S}) = 1\big] - \Pr\big[\mathsf{D}(\mathsf{R}) = 1\big],$$

where $\mathsf{D}(\mathsf{R})$ denotes the random variable of the distinguisher's output when interacting with R.

The reduction relaxation is then formalized based on a function ϵ that maps distinguishers to their respective performance in $[0,1]$, where $\epsilon(\mathsf{D})$ typically refers to the winning probability of a modified distinguisher D' (the reduction) on the underlying computational problem.

Definition 2.2.9. Let ϵ be a function that maps distinguishers to a value in $[0,1]$. Then, the induced relaxation on a resource R, denoted R^ϵ, is defined as

$$\mathsf{R}^\epsilon \coloneqq \left\{ \mathsf{S} \;\middle|\; \forall \mathsf{D} : |\Delta^{\mathsf{D}}(\mathsf{R},\mathsf{S})| \leq \epsilon(\mathsf{D}) \right\}.$$

We call such a relaxation generally an *ϵ-relaxation* or reduction relaxation.

The ϵ-relaxation satisfies several natural properties, such as the errors just adding up as expressed by the following theorem.

Theorem 2.2.10. *Let $\mathcal{R}$ be an arbitrary specification, and let ϵ_1 and ϵ_2 be arbitrary ϵ-relaxations. Then we have $\left(\mathcal{R}^{\epsilon_1}\right)^{\epsilon_2} \subseteq \mathcal{R}^{\epsilon_1+\epsilon_2}$.*

Proof. This follows directly from the triangle inequality of the distinguishing advantage. □

Moreover, they naturally commute with protocol application and parallel composition of additional resources, i.e., the relaxation can be "pulled out". In such a step, however, the additional resource or converter has to be explicitly accounted for in the reduction.

Theorem 2.2.11. *The ϵ-relaxation is* compatible *with protocol application in the following sense:*

$$\boldsymbol{\pi}\left(\mathcal{R}^\epsilon\right) \subseteq \left(\boldsymbol{\pi}\mathcal{R}\right)^{\epsilon_{\boldsymbol{\pi}}},$$

for $\epsilon_{\boldsymbol{\pi}}(\mathsf{D}) \coloneqq \epsilon(\mathsf{D}\boldsymbol{\pi}(\cdot))$, where $\mathsf{D}\boldsymbol{\pi}(\cdot)$ denotes the distinguisher that first attaches $\boldsymbol{\pi}$ to the given resource and then executes D. Moreover, the ϵ-relaxation is compatible with parallel composition, i.e.,

$$\left[\mathcal{R}^\epsilon, \mathcal{S}\right] \subseteq \left[\mathcal{R}, \mathcal{S}\right]^{\epsilon_{\mathcal{S}}},$$

for $\epsilon_{\mathcal{S}}(\mathsf{D}) \coloneqq \sup_{\mathsf{S}\in\mathcal{S}} \epsilon(\mathsf{D}[\,\cdot\,,\mathsf{S}])$, where $\mathsf{D}[\,\cdot\,,\mathsf{S}]$ denotes the distinguisher that emulates S in parallel to the given resource and then lets D interact with them.

Proof. Let $\mathsf{S} \in \boldsymbol{\pi}(\mathcal{R}^\epsilon)$ be arbitrary. Then, by definition, there exists a $\mathsf{T} \in \mathcal{R}^\epsilon$ such that $\mathsf{S} = \boldsymbol{\pi}\mathsf{T}$. Observe that $\mathsf{T} \in \mathcal{R}^\epsilon$ directly implies $|\Delta^{\mathsf{D}}(\mathsf{R}, \mathsf{S})| \leq \epsilon$, and thus

$$|\Delta^{\mathsf{D}}(\boldsymbol{\pi}\mathsf{R}, \mathsf{S})| = |\Delta^{\mathsf{D}}(\boldsymbol{\pi}\mathsf{R}, \boldsymbol{\pi}\mathsf{T})| = |\Delta^{\mathsf{D}\boldsymbol{\pi}(\cdot)}(\mathsf{R}, \mathsf{T})| \leq \epsilon(\mathsf{D}\boldsymbol{\pi}(\cdot)) = \epsilon_{\boldsymbol{\pi}}(\mathsf{D}),$$

implying $\mathsf{S} \in (\boldsymbol{\pi}\mathcal{R})^{\epsilon_{\boldsymbol{\pi}}}$, i.e., compatibility with protocol application.

Now, let $\mathsf{U} \in \left[\mathcal{R}^\epsilon, \mathcal{S}\right]$ be arbitrary. Then, by definition there exists $\mathsf{V} \in \mathcal{R}^\epsilon$ and $\mathsf{S} \in \mathcal{S}$ such that $\mathsf{U} = [\mathsf{V}, \mathsf{S}]$. Moreover, by definition there exists $\mathsf{R} \in \mathcal{R}$ such that $|\Delta^{\mathsf{D}}(\mathsf{R}, \mathsf{V})| \leq \epsilon$. As a consequence we obtain

$$\begin{aligned}|\Delta^{\mathsf{D}}([\mathsf{R}, \mathsf{S}], \mathsf{U})| = |\Delta^{\mathsf{D}}([\mathsf{R}, \mathsf{S}], [\mathsf{V}, \mathsf{S}]| = |\Delta^{\mathsf{D}[\,\cdot\,, \mathsf{V}]}(\mathsf{R}, \mathsf{V})| \\ \leq \epsilon(\mathsf{D}[\,\cdot\,, \mathsf{V}]) \leq \epsilon_{\mathcal{S}}(\mathsf{D}),\end{aligned}$$

and thus $\mathsf{U} \in [\mathcal{R}, \mathcal{S}]^{\epsilon_{\mathcal{S}}}$, concluding the proof. □

2.2.5 Two Important Special Cases

Simulation-based security. In many cases, one considers constructions where the ideal specification exhibits a specific structure. First, most specifications considered in this thesis involve a so-called simulator $\boldsymbol{\sigma}$. That is, a protocol attached to the interfaces controlled by the adversary that "translates" the attacks from the real-world adversary to the ideal world such that they achieve the same effect. Expressing the specification $\boldsymbol{\sigma}\mathcal{S}$ in such a decomposed manner is often beneficial as it makes the achieved security properties obvious and abstracts away other details. For instance, the specification of confidential channels can then be written as the specification $\mathcal{S}$ of channels that only leak the message length, combined with an arbitrary simulator (from some set of simulators under consideration). From this description it is then immediately apparent that for any resource in the combined specification $\boldsymbol{\sigma}\mathcal{S}$ the adversary does not learn more about the message than the length.

Second, for most parts of this thesis we are also interested in computational security, i.e., consider an ideal specification with an ϵ-relaxation applied. In short, we primarily focus on specifications of the form $(\boldsymbol{\sigma}\mathcal{S})^\epsilon$, combining those two aspects. As a result, we introduce the following short-hand notation.

Definition 2.2.12. Let $\mathcal{R}$ and $\mathcal{S}$ be arbitrary specifications, let $\boldsymbol{\pi}$ be an arbitrary protocol for $\mathcal{R}$, $\boldsymbol{\sigma}$ an arbitrary simulator for $\mathcal{S}$, and let ϵ be a function that maps distinguishers to a value in $[0,1]$. Then, we define

$$\mathcal{R} \overset{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon}{\underset{\mathsf{sim}}{\longmapsto}} \mathcal{S} \;:\Longleftrightarrow\; \boldsymbol{\pi}\mathcal{R} \subseteq (\boldsymbol{\sigma}\mathcal{S})^{\epsilon},$$

and say that the protocol $\boldsymbol{\pi}$ constructs $\mathcal{S}$ from $\mathcal{R}$ within ϵ and with respect to the simulator $\boldsymbol{\sigma}$.

For this special type of constructions, the following composition theorem can be derived.

Corollary 2.2.13. *For any specifications $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$, any protocols $\boldsymbol{\pi}$ and $\boldsymbol{\pi'}$, any simulators $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma'}$, and any ϵ-relaxation ϵ and ϵ', we have*

1. $\mathcal{R} \overset{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon}{\underset{\mathsf{sim}}{\longmapsto}} \mathcal{S} \wedge \mathcal{S} \overset{\boldsymbol{\pi'},\boldsymbol{\sigma'},\epsilon'}{\underset{\mathsf{sim}}{\longmapsto}} \mathcal{T} \implies \mathcal{R} \overset{\boldsymbol{\pi'}\circ\boldsymbol{\pi},\boldsymbol{\sigma}\circ\boldsymbol{\sigma'},\epsilon'_{\boldsymbol{\sigma}}+\epsilon_{\boldsymbol{\pi'}}}{\underset{\mathsf{sim}}{\longmapsto}} \mathcal{T}$,
2. $\mathcal{R} \overset{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon}{\underset{\mathsf{sim}}{\longmapsto}} \mathcal{S} \implies [\mathcal{R},\mathcal{T}] \overset{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon_{\mathcal{T}}}{\underset{\mathsf{sim}}{\longmapsto}} [\mathcal{S},\mathcal{T}]$,

where $\epsilon_{\boldsymbol{\pi'}}$, $\epsilon'_{\boldsymbol{\sigma}}$ and $\epsilon_{\mathcal{T}}$ are defined as in Theorem 2.2.11, respectively.

Proof. For the first property, observe that

$$\begin{aligned}\boldsymbol{\pi'}(\boldsymbol{\sigma}\mathcal{S})^{\epsilon} \subseteq (\boldsymbol{\pi'}\boldsymbol{\sigma}\mathcal{S})^{\epsilon_{\boldsymbol{\pi'}}} = (\boldsymbol{\sigma}\boldsymbol{\pi'}\mathcal{S})^{\epsilon_{\boldsymbol{\pi'}}} &\subseteq \left(\boldsymbol{\sigma}(\boldsymbol{\sigma'}\mathcal{T})^{\epsilon'}\right)^{\epsilon_{\boldsymbol{\pi'}}} \\ &\subseteq \left((\boldsymbol{\sigma}\boldsymbol{\sigma'}\mathcal{T})^{\epsilon'_{\boldsymbol{\sigma}}}\right)^{\epsilon_{\boldsymbol{\pi'}}} \subseteq (\boldsymbol{\sigma}\boldsymbol{\sigma'}\mathcal{T})^{\epsilon'_{\boldsymbol{\sigma}}+\epsilon_{\boldsymbol{\pi'}}}\end{aligned}$$

where the first step follows from Theorem 2.2.11, the second step from composition-order invariance, the third step from the assumption and Proposition 2.2.7, the fourth step from Theorem 2.2.11, and the final step from Theorem 2.2.10. Using that specification containment is preserved under protocol application, i.e.,

$$(\boldsymbol{\pi'} \circ \boldsymbol{\pi})\mathcal{R} = \boldsymbol{\pi'}(\boldsymbol{\pi}\mathcal{R}) \subseteq \boldsymbol{\pi'}(\boldsymbol{\sigma})^{\epsilon},$$

then directly implies the desired result.

For the second property, we have

$$\boldsymbol{\pi}\big[\mathcal{R},\mathcal{T}\big] = \big[\boldsymbol{\pi}\mathcal{R},\mathcal{T}\big] \subseteq \big[(\boldsymbol{\sigma}\mathcal{S})^{\epsilon},\mathcal{T}\big] \subseteq [\boldsymbol{\sigma}\mathcal{S},\mathcal{T}]^{\epsilon_{\mathcal{T}}} = \big(\boldsymbol{\sigma}[\mathcal{S},\mathcal{T}]\big)^{\epsilon_{\mathcal{T}}},$$

where the third step follows from Theorem 2.2.11. □

Asymptotic security. Generally, concrete security treatments exhibit many advantages over asymptotic definitions. Nevertheless—for instance when comparing to existing asymptotic notions—it may be convenient to explicitly consider an asymptotic construction notion.

In the following, it is assumed that some computational model and corresponding efficiency and negligibility notions are fixed. For instance, when comparing against traditional game-based notions, one usually considers Turing machines with the polynomial-time being the efficiency notion. An algorithm A is then said to be efficient if it runs in probabilistic polynomial time (PPT), i.e., A has access to random bits and there is a polynomial p such that $A(x)$ terminates after at most $p(|x|)$ steps for all inputs x. Conversely, a function f is said to be negligible if for every polynomial p, there exists $n_0 \in \mathbb{N}$ such that $f(n) < \frac{1}{p(n)}$ for all $n \geq n_0$.

The system model of Constructive Cryptography, including resources and converters, and the construction notion are inherently non-asymptotic. Nevertheless, asymptotic statements can be obtained by considering asymptotic families of resources $\langle \mathsf{R}_\lambda \rangle_{\lambda \in \mathbb{N}}$. Considering such families, the asymptotic notions of efficiency and negligibility can be naturally adapted.

Definition 2.2.14. Let $\langle \mathsf{R}_\lambda \rangle_{\lambda \in \mathbb{N}}$ be a family of resources indexed by the security parameter λ. We say that $\langle \mathsf{R}_\lambda \rangle_{\lambda \in \mathbb{N}}$ is efficient, if there exists a uniform PPT (taking 1^λ as input) that implements it, and denote by Θ_{poly} the set of all such resource families. The notions of efficient protocols and distinguishers are defined analogously.

For a set of efficient resource families $\mathcal{R} \subseteq \Theta_{poly}$ and $\kappa \in \mathbb{N}$, we denote by $\mathcal{R}_\kappa := \{\mathsf{R}_\kappa \mid \langle \mathsf{R}_\lambda \rangle_{\lambda \in \mathbb{N}} \in \mathcal{R}\} \subseteq \Theta$ the resource specification obtained by taking the κ-th element of every family.

Finally, a family of functions $\langle \epsilon_\lambda \rangle_{\lambda \in \mathbb{N}}$ mapping distinguishers to values in $[0, 1]$ is said to be negligible, if for all families of efficient distinguishers $\langle \mathsf{D}_\lambda \rangle_{\lambda \in \mathbb{N}}$, the function $\lambda \mapsto \epsilon_\lambda(\mathsf{D}_\lambda)$ is negligible.

Based on those definitions, we now introduce an asymptotic computational and simulation-based construction notion. This construction notion closely resembles the ones by other composable frameworks, such as the UC framework, that hard-code those aspects.

Definition 2.2.15. Let $\mathcal{R} \subseteq \Theta_{poly}$ and $\mathcal{S} \subseteq \Theta_{poly}$ be two specifications of efficient resource families, and let $\langle \boldsymbol{\pi}_\lambda \rangle_{\lambda \in \mathbb{N}}$ be an efficient protocol family for the the respective $\mathcal{R}_\lambda$. If there exists an efficient simulator family $\langle \boldsymbol{\sigma}_\lambda \rangle_{\lambda \in \mathbb{N}}$, and a negligible family of functions $\langle \epsilon_\lambda \rangle_{\lambda \in \mathbb{N}}$, such that

$$\forall \lambda \in \mathbb{N} : \boldsymbol{\pi}_\lambda(\mathcal{R}_\lambda) \subseteq \left(\boldsymbol{\sigma}_\lambda(\mathcal{S}_\lambda)\right)^{\epsilon_\lambda},$$

we say that the protocol $\boldsymbol{\pi}$ asymptotically constructs $\mathcal{S}$ from $\mathcal{R}$, and denote it by

$$\mathcal{R} \xrightarrow[\mathsf{asympt}]{\langle \boldsymbol{\pi}_\lambda \rangle_{\lambda \in \mathbb{N}}} \mathcal{S}.$$

In a slight abuse of notation, we usually omit the protocol's explicit dependence on the security parameter λ.

As usual, we can derive a more concrete composition theorem for this type of constructions.

Corollary 2.2.16. *For any specifications of efficient families $\mathcal{R}, \mathcal{S}, \mathcal{T} \subseteq \Theta_{poly}$, and any efficient protocol families $\langle \boldsymbol{\pi}_\lambda \rangle_{\lambda \in \mathbb{N}}$ and $\langle \boldsymbol{\pi}'_\lambda \rangle_{\lambda \in \mathbb{N}}$, we have*

1. $\mathcal{R} \xrightarrow[\mathsf{asympt}]{\boldsymbol{\pi}} \mathcal{S} \wedge \mathcal{S} \xrightarrow[\mathsf{asympt}]{\boldsymbol{\pi}'} \mathcal{T} \implies \mathcal{R} \xrightarrow[\mathsf{asympt}]{\boldsymbol{\pi}' \circ \boldsymbol{\pi}} \mathcal{T}$,
2. $\mathcal{R} \xrightarrow[\mathsf{asympt}]{\boldsymbol{\pi}} \mathcal{S} \implies [\mathcal{R}, \mathcal{T}] \xrightarrow[\mathsf{asympt}]{\boldsymbol{\pi}} [\mathcal{S}, \mathcal{T}]$.

Proof (Sketch). This follows directly from Corollary 2.2.13 and the closedness properties of the standard efficiency and negligibility notions. In particular, for two efficient simulator families $\langle \boldsymbol{\sigma}_\lambda \rangle_{\lambda \in \mathbb{N}}$ and $\langle \boldsymbol{\sigma}'_\lambda \rangle_{\lambda \in \mathbb{N}}$, their sequential composition is efficient as well. Moreover, if $\langle \epsilon_\lambda \rangle_{\lambda \in \mathbb{N}}$ is negligible and $\langle \boldsymbol{\pi}'_\lambda \rangle_{\lambda \in \mathbb{N}}$ and $\mathcal{T} \subseteq \Theta_{poly}$ are efficient, then $\langle (\epsilon_\lambda)_{\boldsymbol{\pi}'_\lambda} \rangle_{\lambda \in \mathbb{N}}$ and $\langle (\epsilon_\lambda)_{\mathcal{T}_\lambda} \rangle_{\lambda \in \mathbb{N}}$ are negligible as well. □

2.2.6 An Example

In this section, we exemplify our notation and conventions by a simple construction statement.

To this end, we consider the construction of a secure communication channel from an authentic one and a shared secret key, using a MAC scheme. The construction thereby assumes that both the sender and the receiver are honest, while an eavesdropper might try learn information

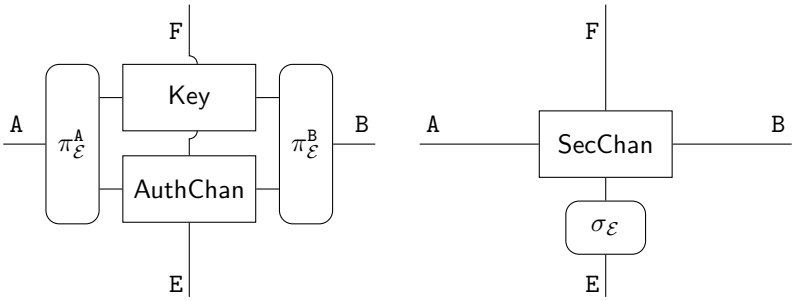


Figure 2.1: Execution of the protocol in the real world by Alice and Bob (left), and the ideal world with the simulator (right). We depict resources using rectangular boxes and converters using rounded boxes.

about the messages sent. Hence, we use the so-called Alice-Bob-Eve setting that considers the parties A (the honest sender Alice), B (the honest receiver Bob), and E (the eavesdropper Eve). In addition, we consider a so-called free interfaces F that are neither controlled by the honest nor dishonest parties, but are rather directly controlled by the environment. In our example, the free interface is useful to express the protocol's correctness condition by enforcing that the honest receiver actually outputs the correct message whenever the environment delivers the corresponding message-tag pair, irrespective of the dishonest party's actions. In summary, we consider a setting with $\mathcal{P} = \{\mathtt{A}, \mathtt{B}, \mathtt{E}, \mathtt{F}\}$, where protocol converters are attached to the interfaces controlled by A and B, respectively, and the simulator is attached to interfaces belonging to E. See Figure 2.1 for an illustration of the construction.

The assumed specification consists of the parallel composition of an authentic channel and a shared secret key, depicted in Figure 2.2. The atomic resources we consider in this work typically only have a single interface per party; thus, in pseudo-code descriptions we just label it by the party's name. Note that the (overly idealized) shared secret key resource does not provide Eve any capabilities, such as influencing the key's availability. For simplicity, we thus omit Eve's interface in the description. When considering a composed resource, such as [AuthChan, Key], then we address a party's interfaces by referring to the corresponding atomic resource's name, e.g., interface A of AuthChan.

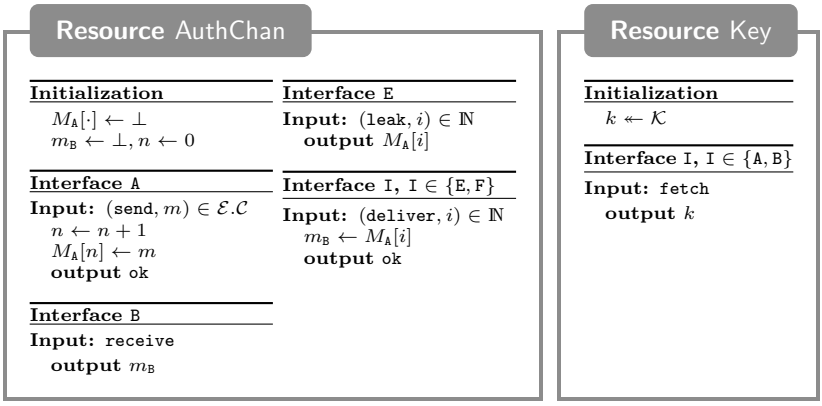


Figure 2.2: The assumed real-world resources of the authenticated-encryption example: an insecure channel and a shared key.

To construct the secure channel, the protocol $\boldsymbol{\pi}_{\mathcal{E}}$ trivially builds on a symmetric encryption scheme $\mathcal{E} = (\mathcal{E}.\mathcal{K}, \mathcal{E}.\mathcal{M}, \mathcal{E}.\mathcal{C}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$, consisting of a key space $\mathcal{E}.\mathcal{K}$, message space $\mathcal{E}.\mathcal{M}$, and ciphertext space $\mathcal{E}.\mathcal{C}$, as well as encryption and decryption algorithms $\mathcal{E}.\mathsf{Enc}$ and $\mathcal{E}.\mathsf{Dec}$, respectively. That is, Alice's converter simply encrypts each message and Bob's one decrypts it, respectively. A description of the converters is presented in Figure 2.3. Note that in a slight abuse of notation, we write a protocol just as a tuple of converters, e.g., $\boldsymbol{\pi}_{\mathcal{E}} \coloneqq (\pi_{\mathcal{E}}^{\mathtt{A}}, \pi_{\mathcal{E}}^{\mathtt{B}})$ instead of explicitly denoting the intended wiring $\gamma_{\mathtt{A}}$ and $\gamma_{\mathtt{B}}$, whenever it is clear from the context. In particular, we typically describe the converters in conjunction with the intended wirings, e.g., stating **call** $(\mathtt{send}, c)$ at int. A of AuthChan in a pseudo-code description.

The constructed specification is described simulation-based, i.e., consists of the secure channel SecChan with a simulator $\sigma_{\mathcal{E}}$ attached to Eve's interface. The secure channel thereby works analogously to the authenticated one, except that Eve's `leak` capability is replaced with a `leakLen` one, returning $|M_{\mathtt{A}}[i]|$. The simulator works straightforwardly by picking its own encryption key, and outputting an encryption of the all-zero string of appropriate length as ciphertext (cf. Figure 2.4a).

It is now easy to see that the construction is secure, if the underlying encryption scheme is IND-CPA secure (and correct).

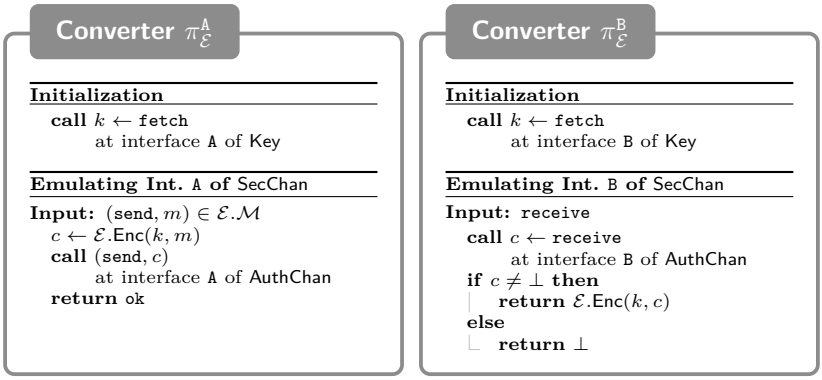


Figure 2.3: The pair of converters using a symmetric encryption scheme $\mathcal{E}$ to construct a secure channel from an authenticated one and a key.

Proposition 2.2.17. *Let $\mathcal{E}$ denote a perfectly correct symmetric encryption scheme, and let* AuthChan, SecChan, Key, $\boldsymbol{\pi}_{\mathcal{E}}$ *and* $\boldsymbol{\sigma}_{\mathcal{E}}$ *be as described above. Moreover, let* CPA_0 *and* CPA_1 *be the two systems defined in Figure 2.4b, formalizing a left-or-right variant of the IND-CPA game. Finally, let* $\boldsymbol{c}$ *denote a reduction protocol that connects to either of the systems and emulates the interface of the assumed or constructed resource as follows:*

- *upon input a message m at Alice's interface, send the pair $(m, 0^{|m|})$ to the connected system and store the returned ciphertext;*
- *upon querying the ciphertext or delivering a message at Eve's interface, return the corresponding ciphertext, or store the corresponding plaintext, respectively;*
- *upon fetching a message at Bob's interface, output the last stored message.*

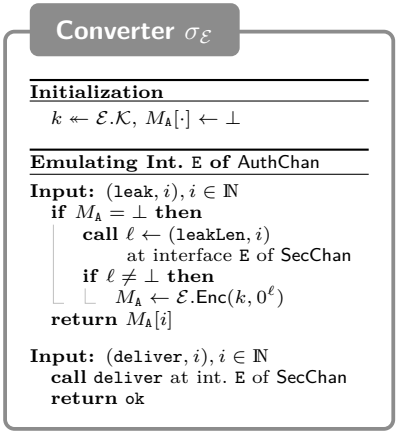
Then, for the reduction $\epsilon(\mathsf{D}) \coloneqq \Delta^{\mathsf{Dc}}(\mathsf{CPA}_0, \mathsf{CPA}_1)$ we have

$$[\mathsf{AuthChan}, \mathsf{Key}] \xrightarrow[\mathsf{sim}]{\boldsymbol{\pi}_{\mathcal{E}}, \boldsymbol{\sigma}_{\mathcal{E}}, \epsilon} \mathsf{SecChan},$$

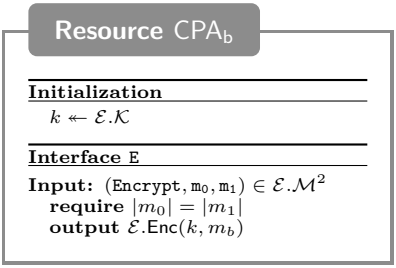
where $\mathsf{D}\boldsymbol{c}$ *denotes the modified distinguisher that first applies* $\boldsymbol{c}$ *to the resource it interacts with.*

Proof. This follows from observing $\boldsymbol{c}\mathsf{CPA}_0 = \boldsymbol{\pi}_{\mathcal{E}}[\mathsf{AuthChan}, \mathsf{Key}]$ and $\boldsymbol{c}\mathsf{CPA}_1 = \boldsymbol{\sigma}_{\mathcal{E}}\mathsf{SecChan}$, where Eve's output matches by definition and Bob's outputs matches by correctness of the scheme. □

Note that in another slight abuse of notation, we allow ourselves to just write a resource R instead of a singleton specification $\{\mathsf{R}\}$ in construction statements.



(a) The simulator.



(b) The two resources for the left-or-right formalization of the IND-CPA game.

Figure 2.4: A formal description of the simulator and the IND-CPA game involved in the construction.

Chapter 3

Constructive Cryptography with Events

In this chapter, we introduce an extension of Constructive Cryptography, which we call *Constructive Cryptography with events*. This extension enhances the abstraction boundary of the modularization units—i.e., resources and converters—by a so-called global event history and, thus, addresses the problem of modularization impeded by dependencies. On a formal level, CC with events primarily consists of an extended system algebra compared to the one introduced in Section 2.2.2. As this novel system algebra satisfies the same higher-level axioms of Constructive Cryptography, results proven at a those abstraction levels directly translate over.

3.1 Introduction

3.1.1 Motivation

At the heart of every composable cryptographic framework, lies the concept of a module or component that provides provides some well-defined interface to the environment. In the traditional Constructive

Cryptography framework, as well as most other composable frameworks, this interface consists solely of the input-output behavior, i.e., components can interact with one another by exchanging by providing explicit inputs and receiving corresponding outputs.

While having such a well-defined abstraction boundary is crucial for modularity, it implies that in certain applications seemingly natural modules cannot be readily modeled as separate components, due to certain interdependencies not appearing as part of their regular communication. For instance, in Chapter 6, we consider the case of secure messaging protocols. In those protocols, it is common that for instance the leakage of one parties memory (one module) can affect the security of a communication channel (another module). Yet, requesting the memory resource to explicitly inform the channel resource about the leakage not only seems unnatural, but is in the traditional system model of Constructive Cryptography outright impossible. As a consequence, one would have to model the memory and communication channel as one joint resource.

In summary, the abstraction boundary imposed by the regular Constructive Cryptography system algebra frequently impedes modularization, resulting in unnecessary complicated and hard to reuse statements.

3.1.2 Contributions

In this chapter, we enhance the interface of resources and converters to relax the the abstraction boundary in a clean and well-controlled manner. More concretely, introduce the notion of global events. Each module (resource or converter) can, thereby, trigger events from a predefined set, on whose occurrence and global order other modules can depend on. This allows the designer of a resource to choose which events are exported as part of the abstraction interface and, thus, greatly enhance the flexibility of the resource paradigm.

This flexibility provides several key advantages. First, it provides a solution to the aforementioned lack of modularization. In Chapter 6, we make use of the global history to properly decompose the statements. For instance, in the above example, the memory resource can export the event of having been leaked, on which the behavior of the channel can then depend on, without this information having to be routed explicitly. Second, and of independent interest, global events also allow to define

novel relaxations. In Chapter 5, we build on global events to propose a novel solution to circumvent the so-called simulator commitment problem of composable security notions.

3.1.3 Related Work

The global event history is somewhat reminiscent of the "directory" ITI used in the recent version (as of December 2018) of the UC framework [Can01] to keep track of corrupted parties. Our notion is, however, more general, as it not only considers corruptions but arbitrary events. Moreover, we stress that our extension is cleaner from a conceptual point of view, as it consists of an alternative instantiation of Constructive Cryptography's higher-level axioms, taking advantage of CC's top-down abstraction approach.

3.2 Systems with Events

3.2.1 The Global Event History

We model events as a generalization of monotone binary outputs (MBO) introduced by Maurer et al. [MPR07]. Roughly, an MBO of a resource is an additional output that can change from 0 to 1 but not back. This can be interpreted as a single event, which happens when the MBO changes to 1. We generalize this to many events by the means of a global event history.

Definition 3.2.1. Let $\mathcal{N}$ be a name set. The *global event history* $\mathcal{E}$ is a list of elements of $\mathcal{N}$ without duplicates.

For $n \in \mathcal{N}$, we use $\mathcal{E}_n$ as a short-hand notation to denote that n is in the list $\mathcal{E}$, and say that the event happened. Analogously, we use $\neg\mathcal{E}_n$ to denote the complementary case. Furthermore, we denote by $\mathcal{E} \xleftarrow{+} \mathcal{E}_n$, the act of appending n to the list $\mathcal{E}$, if $\neg\mathcal{E}_n$, and leaving the list unchanged otherwise.

We also introduce the natural happened-before relation on the events.

Definition 3.2.2. For $n_1, n_2 \in \mathcal{N}$, we say that the event n_1 precedes the event n_2 in the event history $\mathcal{E}$, denoted $\mathcal{E}_{n_1} \prec \mathcal{E}_{n_2}$, if either

- both events happened, i.e, $\mathcal{E}_{n_1}$ and $\mathcal{E}_{n_2}$, and n_1 is in the history before n_2,
- or only n_1 happened so far.

Note that saying that $\mathcal{E}_{n_1} \prec \mathcal{E}_{n_2}$ is true if so far only the former one has happened best matches the type of statement we usually want to make: for instance, if we express the condition that a message is secure if the key has been securely erased before the memory was leaked, then we do not need to insist that the memory actually leaked.

Composite events. An event is essentially just a named monotone condition. Thus, any monotone predicate $P(\mathcal{E})$, e.g. $P(\mathcal{E}) = \mathcal{E}_{n_1} \vee \mathcal{E}_{n_2}$, of the event history can be seen as an event as well. As a consequence, we overload the notation for events accordingly, and for instance write $\mathcal{E}_{n_{12}} \coloneqq \mathcal{E}_{n_1} \vee \mathcal{E}_{n_2}$ instead.

3.2.2 Event-Aware Systems

We consider resources and converters that can (1) read the global event history, and (2) append to the event history from a fixed subset of $\mathcal{N}$.

Definition 3.2.3. An *event-aware resource* R *with associated event-set* $\mathcal{N}_\mathsf{R} \subseteq \mathcal{N}$ is a resource with alphabet $\mathcal{X}' \coloneqq \mathcal{X} \times 2^{\mathcal{E}}$, where $2^{\mathcal{E}}$ denotes the set of all event histories, such that

- for all $i \in \mathbb{N}$, $\mathcal{E}_{X_i}$ is a prefix of $\mathcal{E}_{Y_i}$, and all additional elements thereof are from $\mathcal{N}_\mathsf{R}$, where $\mathcal{E}_{X_i}$ and $\mathcal{E}_{Y_i}$ denote the state of the event history encoded as part of the i-th input and output, respectively;
- the resource is only defined if $\mathcal{E}_{Y_{i-1}}$ is a prefix of $\mathcal{E}_{X_i}$, and all additional elements are *not* from $\mathcal{N}_\mathsf{R}$.

We denote the set of all event-aware systems (with some associated event set) by $\Theta_\mathcal{E}$.

Analogous to regular resources, the parallel composition operator is defined. Besides the party's interface sets having to be disjoint, however, also the associated event-sets have to be disjoint. For two event-aware

resources R and S with disjoint event-sets $\mathcal{N}_{\mathsf{R}}$ and $\mathcal{N}_{\mathsf{S}}$, the resulting resource $[\mathsf{R}, \mathsf{S}]$ has the associated event-set $\mathcal{N}_{\mathsf{R}} \cup \mathcal{N}_{\mathsf{S}}$. Event-aware converters (and hence protocols) are defined analogously as well. We denote by $\Sigma_{\mathcal{E}}$ the set of all event-aware protocols. Note that attaching a converter to a resource is only defined, if their respective event-sets are disjoint, and say that the converter is *compatible* with the resource. Observe that composition order invariance, i.e., Proposition 2.2.3, still holds. It is not affected by the additional conditions imposed by event-awareness.

We use the following conventions. When describing resource and converters (as pseudo-code), we do not treat the event history as part of the input and output, but rather treat $\mathcal{E}$ as a global object. That is, we consider the global event history is an additional component that models event-awareness in an abstract manner, rather than as inputs and outputs that need to be explicitly passed between components. Moreover, to simplify the associated event-set, we use as event-name pairs (id, label), where label is a descriptive keyword (e.g., leaked), and id identifies the resource triggering the event. We then use the notation $\mathcal{E}_{id}^{\mathsf{label}}$.

Finally, note by convention converters implementing protocols do not depend on the event history, since an event formalizes something that *might* be observable, rather than something that is guaranteed to be observable by the honest parties. Event-aware converters are, thus, mainly reserved to simulators.

3.3 Constructions and Relaxations

Since event-aware systems are a special case of the regular ones—with certain additional restrictions on when the operators are defined—the regular construction notion still applies. In particular, the composition theorem of the basic construction notion is preserved as well, which solely builds on composition order invariance. In the following, we describe some differences that apply to more specific types of specifications and their respective construction notions. Note that in terms of notation, we do not make any differences between regular CC and CC with events for constructions and relaxations. For each part of this work, we however make explicit which version of the framework we use.

3.3.1 Event-Aware Reductions

Computational security is defined via a so-called distinguisher, which is a special type of environment, interacting with either of the resources. In Constructive Cryptography, this is formalized using an appropriate ϵ-relaxation (cf. Section 2.2.3).

Clearly, the type of distinguisher used thereby has to be adjusted accordingly when dealing with event-aware resources. Formally, a compatible distinguisher has to provide the current event history with each input, subject to the conditions for the resource to be well-defined.

Definition 3.3.1. For two resources R and S, a distinguisher $\mathbf{D}$ is *compatible* if

- with each input, it provides an event list which is an extension of the one last output by the resource it is interacting with;
- it neither triggers events that are associated with either R or S.

The distinguishing advantage is then accordingly only defined for compatible distinguishers, and the ϵ-relaxation only quantifies over compatible distinguishers.

3.3.2 Event Renaming

The goal of a construction step $\boldsymbol{\pi}\mathcal{R} \subseteq \mathcal{S}$ is to abstract away certain details of the real-world specification $\boldsymbol{\pi}\mathcal{R}$ by the simpler to understand specification $\mathcal{S}$. Introducing global events in a naive way, however, can thwart this goal. For instance, consider a real-world resource containing a leakable memory resource that is used by the protocol to store a key. In the ideal world, this memory might well be abstracted away or replaced by a key. Hence, for the sake of understandability, the memory-leaked event in the real world should be renamed to key-leaked. That is, in the end event's name are just a syntactical placeholder without any deeper meaning, and thus renaming should be applied whenever it aids understandability. Moreover, in certain cases it also seems desirable to have a more fine-grained consideration of events in the ideal world, for instance by separating a message-received event into a successful and unsuccessful one (where the notion of successful might not have been present in the assumed specification but only introduced by the

protocol). In many instances, this allows to formalize the achieved security properties in a more intuitive and simpler manner.

To accommodate such renamings and refinements, we introduce an explicit (partial) event mapping function τ as follows.

Definition 3.3.2. Let R be a resource with associated event set $\mathcal{N}_{\mathsf{R}}$ and $\tau\colon \mathcal{N}_{\mathsf{R}} \to \mathcal{N}$ a (partial) renaming function. Then, we denote by $\tau(\mathsf{R})$ the resource with associated event set $\mathrm{img}(\tau) = \tau(\mathcal{N}_{\mathsf{R}})$ that behaves as follows:

- In terms of input-output behavior, it behaves equivalently to R.
- Upon every output, it applies τ to the global event history (which formally is encoded as part of the output), leaving elements outside $\mathcal{N}_{\mathsf{R}}$ unchanged. Duplicates are dropped if necessary. The unmodified version is stored for handling the next input.
- Upon every input, it undoes τ to the global event history, using the recalled list of modifications.

When referring to real-world events for specifying ideal-world guarantees, we will sometimes use $\tilde{\mathcal{E}} \coloneqq \tau(\mathcal{E})$ as a shorthand notation to denote the event-history as visible to the outside world.

While an event mapping is not a relaxation (a mapped specification is not a superset of the original one), it exhibits similar properties.

Proposition 3.3.3. *Let R be a resource with associated event set $\mathcal{N}_{\mathsf{R}}$. For every protocol $\boldsymbol{\pi}$ and resource S that do not depend on events renamed by τ, we have*

$$\boldsymbol{\pi}\tau(\mathsf{R}) = \tau(\boldsymbol{\pi}\mathsf{R})$$

and

$$[\tau(\mathsf{R}), \mathsf{S}] = \tau([\mathsf{R}, \mathsf{S}]).$$

Proof. This follows directly from the definition of $\tau(\mathsf{R})$. □

Since we mostly consider protocols that are event agnostic, this in particular implies that event mapping interacts gracefully with the construction notion, as expressed by the following corollary.

Corollary 3.3.4. *Let $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$ be specifications and $\boldsymbol{\pi}$ and $\boldsymbol{\pi'}$ protocols. If for the first property $\boldsymbol{\pi'}$ and for the second property $\mathcal{T}$ do not depend on events renamed by τ, then we have*

1. $\mathcal{R} \overset{\boldsymbol{\pi}}{\longmapsto} \tau(\mathcal{S}) \wedge \mathcal{S} \overset{\boldsymbol{\pi'}}{\longmapsto} \tau'(\mathcal{T}) \implies \mathcal{R} \overset{\boldsymbol{\pi'} \circ \boldsymbol{\pi}}{\longmapsto} \tau(\tau'(\mathcal{T}))$,

2. $\mathcal{R} \overset{\boldsymbol{\pi}}{\longmapsto} \tau(\mathcal{S}) \implies \big[\mathcal{R}, \mathcal{T}\big] \overset{\boldsymbol{\pi}}{\longmapsto} \tau(\big[\mathcal{S}, \mathcal{T}\big])$.

Proof. The properties follow directly from Proposition 3.3.3, as

$$\boldsymbol{\pi'}(\boldsymbol{\pi}\mathcal{R}) \subseteq \boldsymbol{\pi'}(\tau(\mathcal{S})) = \tau(\boldsymbol{\pi'}\mathcal{S}) \subseteq \tau(\tau'(\mathcal{T}))$$

and

$$\boldsymbol{\pi}[\mathcal{R}, \mathcal{T}] = [\boldsymbol{\pi}\mathcal{R}, \mathcal{T}] \subseteq [\tau(\mathcal{S}), \mathcal{T}] = \tau([\mathcal{S}, \mathcal{T}]). \qquad \square$$

For simulation-based constructions, i.e., $\mathcal{R} \overset{\boldsymbol{\pi}, \boldsymbol{\sigma}, \epsilon}{\underset{\mathsf{sim}}{\longmapsto}} \mathcal{S}$, the analogous result can be stated, as long as $\boldsymbol{\sigma}$ does not depend on events renamed by τ. In all situations encountered in this thesis, this condition holds true.

While the event mapping could be made a further parameter of the construction syntax, we refrain from doing so due to the rare usage. Rather, we just state the remapping in conjunction with the construction. Observe that in the case of simulation-based constructions, the event remapping actually refers to the entire ideal specification $(\boldsymbol{\sigma}\mathcal{S})^{\epsilon}$ rather than just $\mathcal{S}$.

Chapter 4

Context-Restricted Constructions

In this chapter, we study the pitfall of popular and powerful abstraction boundaries that are known to be impossible to construct from resources readily available in the real world, and propose the notion of context-restricted constructions to salvage them.

4.1 Introduction

4.1.1 Motivation

One of the obstacles that composable frameworks face, is that sometimes widely used abstraction boundaries are known to be impossible to construct from resources readily available in the real world.

An illuminating example thereof is the so-called random oracle [BR93], which originally was introduced as an powerful abstraction of hash functions. Over the years, the random oracle model (ROM) has proven to be an important tool towards establishing confidence in the security of real-world cryptographic constructions. The paradigm can be described in two steps: first, to design a protocol and prove it secure in the ROM, thus using a random oracle instead of a hash function; second, to instantiate the random oracle with a cryptographic hash

function. The random oracle is, however, not only used as a model to prove protocols in, but it also serves as a reference point for the designers of hash functions. The indifferentiability framework [MRH04], while being a general framework, is most famously used to phrase the security obligation of a hash function construction: the hash function is proven indifferentiable from a random oracle when using an ideal compression function (e.g. a fixed input-length random oracle), thereby excluding attacks exploiting the construction. Since indifferentiability is equipped with a composition theorem, this guarantee holds moreover irrespective of the context the hash function is used in.

However, it is well known [CGH04] that no hash function realizes a random oracle; hence, once the random oracle is instantiated the security proof degenerates to a heuristic security argument. Analogously, just as no hash function can instantiate a random oracle, no real compression function can instantiate the idealized version assumed in the proof. In the language of Constructive Cryptography, this means that for a public random oracle resource PO and a public hash key resource HK, there exists no deterministic and stateless protocol $\boldsymbol{h}$ (implementing a hash function), such that $\mathsf{HK} \xrightarrow[\mathsf{asympt}]{\boldsymbol{h}} \mathsf{PO}$. Nevertheless, most real-world protocols built on the random oracle paradigm are believed to be secure and indifferentiability proofs are still considered vital towards establishing confidence in hash function constructions—despite the unsoundness of the general paradigm.

To resolve this apparent contradiction between the ROM paradigm proven to be unsound and actual protocols building on it believed to be secure, Bellare, Hoang, and Keelveedhi [BHK14] proposed the notion of *universal computational extractors (UCE)*. This notion is based on the observation that for most "real-world" protocols proven secure in the random oracle model, instantiating the random oracle with a concrete hash function is not known to be insecure. The UCE framework revisits the question of what it means for a hash functions to "behave like a random oracle" and formalizes families of security notions aimed at bridging the gap between the general impossibility result and the apparent security of concrete protocols. UCE is, however, formalized as a multi-stage game without clear composition guarantees. In fact, Ristenpart, Shacham, and Shrimpton pointed out [RSS11], that multi-stage games generally interact poorly with composable security

definitions, invalidating some of the intuitive properties one might assume. This raises the question, how one can soundly embed a UCE-style definition in a composable framework, such that the resulting notion has clear semantics and preserves (most of) the benefits of composable security frameworks.

4.1.2 Universal Computational Extractors

Bellare, Hoang, and Keelveedhi [BHK13a] proposed the notion of *universal computational extractors (UCE)*, refining the predominant random oracle methodology. This notion is based on the observation that for most "real-world" protocols proven secure in the random oracle model, instantiating the random oracle with a concrete hash function is not known to be insecure. The UCE framework revisits the question of what it means for a hash functions to "behave like a random oracle" and formalizes families of security notions aimed at bridging the gap between the general impossibility result for the ROM and the apparent security of concrete protocols.

The UCE framework defines a two-stage adversary, where only the first stage—the *source* S—has access to the oracle (either the hash function or the random oracle) and only the second stage—the *distinguisher* D—has access to the hash key hk. The source provides some *leakage* L to the distinguisher that then decides with which system the source interacted. The definition of the security game $\mathsf{UCE}_H^{S,D}$ is presented in Figure 4.1. Therein, H.Kg denotes the key-generation algorithm, H.Ev the deterministic evaluation algorithm, and l the output length associated with the family of hash functions H. Finally,

$$\mathbf{Adv}^{\mathrm{uce}}_{H,S,D}(\lambda) \coloneqq 2\Pr[\mathsf{UCE}_H^{S,D}(\lambda) = 1] - 1$$

denotes the UCE-advantage of (S, D) on H.

Without any further restriction, this game is trivial to win: the source queries some point x, obtains the result y, and then provides the tuple (x, y) as leakage to the distinguisher which then decides whether y matches with the hash of x. Therefore, in order for this definition to be meaningful, the leakage has to be restricted in some sense, which gives rise to various *UCE-classes* depending on the kind of restriction. The basic restriction proposed was that the queries of the source S must be unpredictable given the leakage L. Both statistical unpredictability as

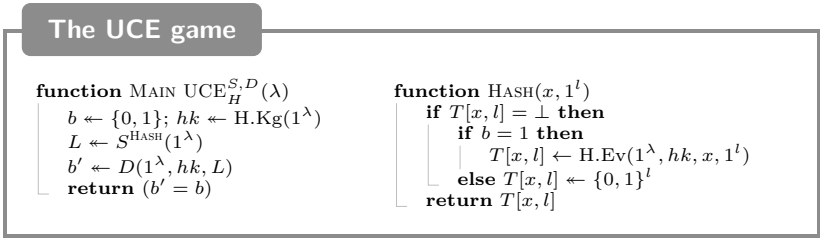


Figure 4.1: The UCE game for a hash function H, a source S, and a distinguisher D.

well as computational unpredictability have been proposed. However, the latter has been shown to be impossible assuming iO exists [BFM14], leading to the proposal of a variety of other computational source families in [BHK13b].

4.1.3 Indifferentiability

In this chapter, we consider the so-called indifferentiability setting. While indifferentiability—originally introduced by Maurer, Renner, and Holenstein [MRH04]—is a general framework, it is most famously used to phrase the security obligation of a hash function construction: the hash function is proven indifferentiable from a random oracle when using an ideal compression function (e.g. a fixed input-length random oracle), thereby excluding attacks exploiting the construction. Indifferentiability is a special case of the Constructive Cryptography framework, as pointed out by Maurer and Renner [MR16]

First, indifferentiability is simulation based. Second, indifferentiability considers a setting with one honest party Alice, and a dishonest party Eve, i.e., $\mathcal{P} = \{A, E\}$. Hence, whenever a protocol converter $\boldsymbol{\pi}$ is considered, it is understood to be connected to Alice's interfaces, and whenever a simulator $\boldsymbol{\sigma}$ is considered, it is understood to be connected to Eve's interfaces. As a convention, in figures, such as Figure 4.2, Alice's interfaces are drawn on the left and Eve's on the right of the resource. Analogously, in algebraic expressions we write $\boldsymbol{\pi}\mathcal{R}\boldsymbol{\sigma}$ instead of $\boldsymbol{\sigma}\boldsymbol{\pi}\mathcal{R}$ to indicate to which party's interfaces the respective protocol is applied.

Figure 4.2: The real (left) and the ideal (right) setting considered in indifferentiability. The honest party's interface is depicted on the left, and the dishonest's on the right side.

Third, indifferentiability considers so-called *outbound* resources only, meaning that interaction at one party's interfaces does not influence the other party's interfaces. Finally, for the sake of presentation and making our notions comparable to game-based notions, we consider asymptotic constructions, as introduced in Section 2.2.5.

Note that our results in principle naturally carry over to more general Constructive Cryptography settings, with concrete security and potentially more parties, as long as the dishonest party is fixed. We have opted for the simpler indifferentiability setting here, because our concrete results are centered around hash function constructions.

4.1.4 Contributions

Context-restricted constructions. First, we introduce a generalization of the construction notion called *context-restricted constructions*. This generalization allows us to model that a resource cannot be instantiated in every context but only within a well-specified set of contexts. That is, the notion gives up on the universal quantification of environments by only considering certain admitted protocols for the honest users. While this clearly cedes general composability, we demonstrate that the remaining composition guarantees exactly correspond to the ones intuitively expected from having such an admissible context set.

Generalizing UCE. Second, we consider the random oracle resource. We, thereby, show that the UCE framework can be seen as a special case of context-restricted constructions when applied to the construction of a random oracle by the means of a hash function. Thereby we propose an alternative interpretation of the UCE framework in a traditional single-stage adversary model with well-defined composition guarantees and provide a direct relation between the UCE and the indifferentiability

frameworks, which is a special case of the CC framework.

In addition, we introduce two generalizations of the split-source UCE-class, which cannot be expressed in the original UCE framework without alteration. In fact, those generalizations naturally follow by viewing UCE as a special case of context-restricted constructions.

Moreover, we demonstrate that the generalizations of UCE, such as public-seed pseudorandom permutations (psPRP) by Soni and Tessaro [ST17] and interactive computational extractors (ICE) by Farshim and Mittelbach [FM16], can be expressed as context-restricted constructions of respective resources, as well.

Unifying UCE and indifferentiability. Finally, we propose to consider context-restricted constructions as a methodology to analyze the security of common hash-function constructions. In contrast to indifferentiability—which is commonly used to prove the soundness of hash-function constructions—context-restricted constructions allow us to consider more fine-grained versions of both the assumption on the compression function as well as the guarantee of the constructed hash function. In particular, we demonstrate that context-restricted constructions supersede the aspects of UCE (a multi-stage game to allow for protocols that soundly build on a random oracle) and indifferentiability (a composable definition to allow for sound hash function constructions). In other words, we show that context-restricted constructions can overcome the obstacles pointed out by Ristenpart et al. in combining two such notions.

As a technical result, we investigate the split-security of the Merkle-Damgård scheme and we prove that the constructed hash function is split-secure if the underlying compression function is strong-split secure (as opposed to the usual much stronger assumption of the compression function being a random function) if the hashed message has a sufficient min-entropy density from the distinguisher's point of view.

4.1.5 Related Work

Restricted composability. Restricted composability has been considered by Backes, Dürmuth, Hofheinz, and Küsters in the context of the Reactive Simulatability framework [BDHK06]. Analogous to our notion of context-restricted constructions, the authors give up the universal

quantification of environments that allows for general composability by only considering certain admitted protocols for the honest users. On a technical level, however, their approach significantly differs in how those restrictions are formalized. While context-restriction constructions are simply parametrized in a set of admissible protocols, conditional reactive simulatability formalizes the admissibility using so-called predicates. A predicate thereby restricts the kind and the order of messages on ports of the system under consideration. This, however, does for instance not allow to circumvent the impossibility results for the ROM, as an environment testing whether a hash query is consistent with the answer he computed himself cannot be expressed as such a predicate (which only sees that the hash key has been requested and that a single hash query has been performed).

The incompatibility of UCE and indifferentiability. Several works aimed to address the issue pointed out by Ristenpart, Shacham, and Shrimpton in [RSS11] of multi-stage games, such as UCE, and composable notions, such as indifferentiability, not interacting well.

In particular, Ristenpart, Shacham, and Shrimpton introduced *reset indifferentiability* [RSS11] as a workaround to the composition problems in multi-stage settings they highlighted. In [DGHM13], Demay et al. gave an alternative interpretation of the example used by Ristenpart et al. to demonstrate the issue. They prove that reset indifferentiability is equivalent to indifferentiability with stateless simulators. Moreover, they introduce the notion of resource-restricted indifferentiability, which makes the memory used in the simulator explicit.

Finally, in [Mit14], Mittelbach presents a condition called *unsplittability* on multi-stage games, that allows to show that the composition theorem of indifferentiability can be salvaged for iterative hash function constructions. They formalize a condition that specifies certain multi-stage games, in which the random oracle can be safely instantiated by an iterated hash function based on an idealized compression function. It has, however, not been studied to which extend this applies to the various variants of UCE. In addition, unsplittability does not cover hash function constructions based on a less ideal compression function that satisfies a UCE-like security definition.

In comparison, the notion of context-restricted constructions we introduce in this chapter aims at unifying UCE and composable definitions

under the umbrella of a single framework with explicit composability guarantees.

Generalizing UCE. Since the introduction of the UCE framework, various additional UCE-families, and various generalizations of the framework itself, have been proposed.

In [FM16], Farshim and Mittelbach introduced a generalization of UCE called interactive computational extractors (ICE). Generalizing UCE to interactive scenarios is also one of our contributions. The generalization they propose and the one we propose, however, differ on a very fundamental level and pursue different directions. ICE makes the two stages of the original UCE definition symmetrical where the two stages jointly form the queries, requiring that neither one of them can predict the query. In contrast, we exactly use the asymmetry of UCE to embed it in the traditional indifferentiability setting with one dishonest and one honest party, where naturally the honest party knows the position where it queries the hash function.

In [ST17], Soni and Tessaro introduce the notion of public-seed pseudorandom permutations (psPRP) that are inspired by UCE. In fact, they introduce a generalization of UCE, called public-seed pseudorandomness, of which both psPRP and UCE are instantiations. For their psPRP notion they introduce the unpredictability and reset-security notions analogous to UCE, and moreover they study the relations between psPRP and UCE. In contrast to our notion, their definition is still purely game-based. In Section 4.4, we show that context-restricted constructions are a strict generalization of their notion as well.

Using UCE to analyze hash-function constructions. Bellare et al. [BHK14] have also suggested to use the UCE framework to study the domain extension of a finite input-length random oracle to a UCE secure variable input-length random oracle. Their motivation is based on finding more efficient constructions if they only require the UCE-security of the variable input-length random oracle. In contrast, we consider the domain extension in a setting where we also assume the compression function to be only UCE secure.

4.2 Context-Restricted Constructions

In this section, we introduce our notion of context-restricted constructions, introduce the corresponding composition rules, and demonstrate how this novel notion relates to the regular construction notion.

4.2.1 Modeling Context Restrictions

In this section we formally define the idea of restricting composition. In order to do so, we define a context in which we allow the resource S to be used. A context consists of an auxiliary parallel resource P and some converter $\mathbf{f}$ applied by the honest party. We usually call this converter $\mathbf{f}$ a *filter* to indicate that its goal is to restrict the access to the resource S. To obtain general statements, we consider a *set* of contexts instead of a single one. This set should be general enough to capture many application scenarios but avoid those for which the impossibility is known.

Definition 4.2.1. A context set $\mathcal{C}$ is a subset of $\Sigma \times \Theta$, where Σ denotes the set of all protocols for the honest parties and Θ denotes the set of all resources.

Recall that our goal is to make a modular statement: constructing $\mathcal{S}$ from another specification $\mathcal{R}$ in each of these contexts in $\mathcal{C}$, i.e., finding a single protocol $\boldsymbol{\pi}$ such that $\boldsymbol{\pi}\mathcal{R}$ can instantiate $\mathcal{S}$ in each of these contexts of $\mathcal{C}$. Therefore, the same context appears in both the real and the ideal setting. See Figure 4.3 for an illustration of the distinction problem when fixing a specific context. Quantifying over all contexts of a set leads to the following definition of *context-restricted constructions*.

Definition 4.2.2. Let $\mathcal{C} \subseteq \Sigma \times \Theta$ be a given set of contexts, let $\mathcal{R}$ and $\mathcal{S}$ be two specifications, and let $\boldsymbol{\pi}$ be an arbitrary protocol. Moreover, let $\langle \boldsymbol{\sigma}_C \rangle_{C \in \mathcal{C}}$ be simulators for each context, and let $\langle \epsilon_C \rangle_{C \in \mathcal{C}}$ be functions mapping distinguishers to values in $[0, 1]$. Then, we define

$$\mathcal{R} \xrightarrow[\mathsf{cr}]{\boldsymbol{\pi}, \mathcal{C}, \langle \boldsymbol{\sigma}_C \rangle_{C \in \mathcal{C}}, \langle \epsilon_C \rangle_{C \in \mathcal{C}}} \mathcal{S}$$
$$:\Longleftrightarrow \forall (\mathbf{f}, \mathsf{P}) \in \mathcal{C} : \mathbf{f}[\boldsymbol{\pi}\mathcal{R}, \mathsf{P}] \subseteq \left(\mathbf{f}[\mathcal{S}, \mathsf{P}]\boldsymbol{\sigma}_{(\mathbf{f},\mathsf{P})}\right)^{\epsilon_{(\mathbf{f},\mathsf{P})}},$$

and say that the protocol $\boldsymbol{\pi}$ $\mathcal{C}$-restricted constructs $\mathcal{S}$ from $\mathcal{R}$.

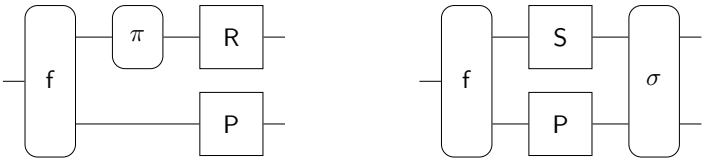


Figure 4.3: The real (left) and the ideal (right) setting considered in the context-restricted construction notion for a specific context $(\mathbf{f}, \mathsf{P})$ consisting of the filter $\mathbf{f}$ and the auxiliary parallel resource P.

For simplicity—and ease of comparing them with the asymptotic UCE notion—we introduce the following asymptotic version of context-restriction constructions.

Definition 4.2.3. Let $\mathcal{C} \subseteq \Sigma_{poly} \times \Theta_{poly}$ be a given set of efficient context families, let $\mathcal{R} \subseteq \Theta_{poly}$ and $\mathcal{S} \subseteq \Theta_{poly}$ be two specifications of efficient resource families, and let $\langle \boldsymbol{\pi}_\lambda \rangle_{\lambda \in \mathbb{N}}$ be an efficient protocol family. If there exists an efficient simulator family $\langle \boldsymbol{\sigma}_{C,\lambda} \rangle_{C \in \mathcal{C}, \lambda \in \mathbb{N}}$, and a negligible family of functions $\langle \epsilon_{C,\lambda} \rangle_{C \in \mathcal{C}, \lambda \in \mathbb{N}}$, such that

$$\forall (\mathbf{f}, \mathsf{P}) \in \mathcal{C}\ \forall \lambda \in \mathbb{N} : \mathbf{f}_\lambda[\boldsymbol{\pi}_\lambda \mathcal{R}_\lambda, \mathsf{P}_\lambda] \subseteq \left(\boldsymbol{\sigma}_{(\mathbf{f},\mathsf{P}),\lambda}\ \mathbf{f}_\lambda[\mathcal{S}_\lambda, \mathsf{P}_\lambda]\right)^{\epsilon_{(\mathbf{f},\mathsf{P}),\lambda}},$$

and say that the protocol $\boldsymbol{\pi}$ asymptotically $\mathcal{C}$-restricted constructs $\mathcal{S}$ from $\mathcal{R}$, and denote it by

$$\mathcal{R} \xvdash[\mathsf{cr\text{-}asym}]{\langle \boldsymbol{\pi}_\lambda \rangle_{\lambda \in \mathbb{N}}, \mathcal{C}} \mathcal{S}.$$

In a slight abuse of notation, we omit in the following the explicit dependence on the security parameter λ.

4.2.2 Composition

Analogous to the basic construction notion introduced in Section 2.2.3, we are interested in deducing certain syntactical derivation rules—called composition rules—for the context-restricted construction notion.

Before stating the composition theorem, we first observe that when a resource S is constructed from R in a context $(\mathbf{f}, \mathsf{P})$, the overall environment of S actually consists of both $(\mathbf{f}, \mathsf{P})$ and the distinguisher.

Especially, if S can be constructed from R within $(\mathbf{f}, \mathsf{P})$, so can it within $(\mathbf{f}' \circ \mathbf{f}, [\mathsf{P}, \mathsf{P}'])$. This is because $\mathbf{f}'$ and P' can be accounted for in the reduction, or in the case of the asymptotic notion be simply absorbed into the distinguisher, respectively. As a consequence, we define the following closure operation on context sets.

Definition 4.2.4. Let $\mathcal{C} \subseteq \Sigma \times \Theta$ be a given set of contexts. We denote by $\bar{\mathcal{C}} \subseteq \Sigma \times \Theta$ the following set of contexts:

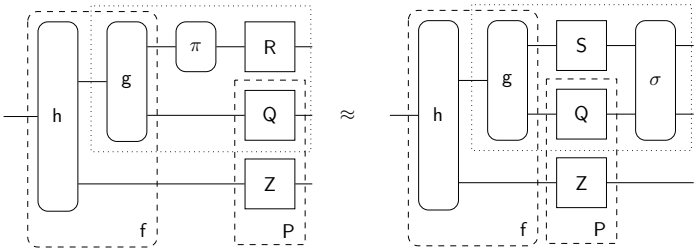
$$\bar{\mathcal{C}} \coloneqq \{(\mathbf{f}, \mathsf{P}) \in \Sigma \times \Theta \mid \exists (\mathbf{g}, \mathsf{Q}) \in \mathcal{C}\ \exists \mathbf{h} \in \Sigma\ \exists \mathsf{U} \in \Theta :\ \mathbf{h} \circ \mathbf{g} = \mathbf{f}\ \wedge\ [\mathsf{Q}, \mathsf{U}] = \mathsf{P}\}.$$

The following proposition states the context-restricted construction notion is idempotent under the closure of the context set.

Proposition 4.2.5. *Let $\mathcal{R}, \mathcal{S}$ denote two specifications of efficient resources, $\boldsymbol{\pi} \in \Sigma$ denote a protocol, and let $\mathcal{C}$ denote a set of contexts. We then have*

$$\mathcal{R} \xmapsto[\text{cr-asym}]{\boldsymbol{\pi}, \mathcal{C}} \mathcal{S} \quad \Longleftrightarrow \quad \mathcal{R} \xmapsto[\text{cr-asym}]{\boldsymbol{\pi}, \bar{\mathcal{C}}} \mathcal{S}.$$

Proof. The implication $\Longleftarrow$ is trivial, since $\mathcal{C} \subseteq \bar{\mathcal{C}}$. We now prove the other direction. Let $(\mathbf{f}, \mathsf{P}) \in \bar{\mathcal{C}}$ and notice that by Definition 4.2.4 this implies that there exists $(\mathbf{g}, \mathsf{Q}) \in \mathcal{C}$, $\mathbf{h} \in \Sigma$, and $\mathsf{Z} \in \Theta$ such that $\mathbf{h} \circ \mathbf{g} = \mathbf{f}$ and $[\mathsf{Q}, \mathsf{Z}] = \mathsf{P}$.



By our assumption, we know that for all $\mathsf{R} \in \mathcal{R}$, there exists a $\mathsf{S} \in \mathcal{S}$ and an appropriate simulator $\boldsymbol{\sigma}$ such that $\mathbf{g}\boldsymbol{\pi}[\mathsf{R}, \mathsf{Q}]$ is computationally indistinguishable from $\mathbf{g}[\mathsf{S}, \mathsf{Q}]\boldsymbol{\sigma}$, as indicated by the dotted box in the above figure. Thus, if we add the additional filter $\mathbf{h}$ and resource Z, they remain indistinguishable. □

We now state the composition theorem of the context-restricted construction notion. Note that the additional conditions compared to the regular composition theorem (cf. Section 2.2.3) are a direct consequence of the context restrictions. For instance, if in the sequential case we construct $\mathcal{T}$ from $\mathcal{S}$ in one of the given contexts, we have to ensure that now we are again in a valid context for constructing $\mathcal{S}$ from $\mathcal{R}$. This highlights that in order for context-restricted constructions to be useful, the context sets have to be defined in a form that containment can be easily verified.

Theorem 4.2.6. *Let $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$ denote specifications, let $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ denote protocols, and $\mathcal{C}_1$ and $\mathcal{C}_2$ contexts sets. We have*

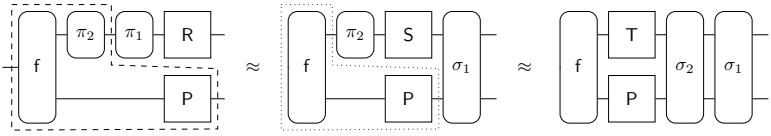
$$\mathcal{R} \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi_1},\mathcal{C}_1} \mathcal{S} \;\wedge\; \mathcal{S} \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi_2},\mathcal{C}_2} \mathcal{T} \quad\Longrightarrow\quad \mathcal{R} \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi_2}\circ\boldsymbol{\pi_1},\mathcal{C}_2} \mathcal{T},$$

iff for all $(\mathbf{f},\mathsf{P}) \in \mathcal{C}_2$ it holds that $(\mathbf{f}\circ\boldsymbol{\pi}_2,\mathsf{P}) \in \overline{\mathcal{C}_1}$. Moreover, we have

$$\mathcal{R} \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi}_1,\mathcal{C}_1} \mathcal{S} \quad\Longrightarrow\quad [\mathcal{R},\mathcal{U}] \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi}_1,\mathcal{C}_2} [\mathcal{S},\mathcal{U}],$$

iff for all $(\mathbf{f},\mathsf{P}) \in \mathcal{C}_2$ it holds that $(\mathbf{f},[\mathcal{U},\mathsf{P}]) \subseteq \overline{\mathcal{C}_1}$.

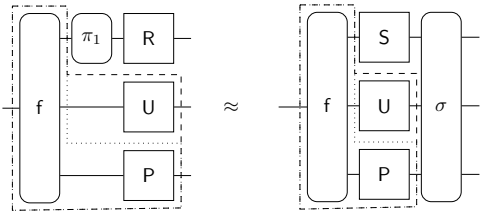
Proof. We first show the sequential case. Assume that the prerequisite regarding the two context sets is satisfied. Moreover, consider an arbitrary context $(\mathbf{f},\mathsf{P}) \in \mathcal{C}_2$ and the three system configurations, depicted in the following figure.



Using the assumed property on $\mathcal{C}_1$ and Proposition 4.2.5, we know that the context indicated with the dashed line is a valid one and thus, for every $\mathsf{R} \in \mathcal{R}$, there exists a $\mathsf{S} \in \mathcal{S}$, such that the first two systems from the above figure are computationally indistinguishable. The second equality follows directly from the premise. For the reverse direction, assume that the implication does not hold, i.e., there exists a context $(\mathbf{f},\mathsf{P})$ such that first and third systems are distinguishable. By assumption we

have, however, that the second and third are indistinguishable. Hence, the first and second system must be distinguishable, which however by assumption implies that the dashed context is not in $\overline{\mathcal{C}_1}$.

In order to show the parallel composition property, assume again that the corresponding condition on the context sets is satisfied. Moreover, consider an arbitrary context $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}_1$ and the two system configurations, depicted in the following figure.



Using the assumed property on $\mathcal{C}_1$ and Proposition 4.2.5, we know that within the context indicated with the dashed line, for every $\mathsf{R} \in \mathcal{R}$ there exists a $\mathsf{S} \in \mathcal{S}$, such that the two systems are indistinguishable. In short, the parallel composition property is just associativity: The resource U can be seen as both part of the context, indicated by the dashed line, or part of the real and ideal resources, indicated by the dotted line. The reverse direction, is again easy to see using an indirect proof of implication. □

4.2.3 The Relation to Regular Constructions

Recall that $\boldsymbol{id}$ denotes the identity protocol for which we have $\boldsymbol{id}\,\mathsf{R} = \mathsf{R}$, and that $\square$ denotes the neutral resource for which we have $[\mathsf{R}, \square] = \mathsf{R}$, for any resource R. It is then easy to see that the regular simulation-based construction notion, which guarantees full composition, is a special case of the context-restricted notion with the context set $\mathcal{C}_{\mathrm{id}} \coloneqq \{(\boldsymbol{id}, \square)\}$, since $\overline{\mathcal{C}_{\mathrm{id}}} = \Sigma \times \Theta$, i.e., the closure equals to the set of all resources and converters. One can, however, also take the opposite point of view and consider context-restricted constructions to be a special case of the plain notion. From this perspective, a context-restricted constructions is just a set of normal construction statements where the context is part of the considered resources and protocols, respectively. This can

be summarized in the following proposition.

Proposition 4.2.7. *Let* $\mathcal{C}_{\mathrm{id}} \coloneqq \{(\boldsymbol{id}, \square)\}$. *For all specifications* $\mathcal{R}$, $\mathcal{S}$, *protocols* $\boldsymbol{\pi}$, *and context sets* $\mathcal{C}$, *we have*

$$\mathcal{R} \xrightarrow[\text{asympt}]{\boldsymbol{\pi}} \mathcal{S} \quad \Longleftrightarrow \quad \mathcal{R} \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi}, \mathcal{C}_{\mathrm{id}}} \mathcal{S},$$

$$\mathcal{R} \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi}, \mathcal{C}} \mathcal{S} \quad \Longleftrightarrow \quad \forall (\mathbf{f}, \mathsf{P}) \in \mathcal{C} \colon \ [\mathcal{R}, \mathsf{P}] \xrightarrow[\text{asympt}]{\mathbf{f} \circ \boldsymbol{\pi}} \mathbf{f}[\mathcal{S}, \mathsf{P}].$$

Proof. This follows directly from Definitions 2.2.15 and 4.2.3, and the definitions of the identity protocol $\boldsymbol{id}$ and the neutral resource $\square$, respectively. □

Using $\overline{\mathcal{C}_{\mathrm{id}}} = \Sigma \times \Theta$, it is also easy to see that the composition theorem Corollary 2.2.16 for regular (asymptotic) simulation-based constructions is just a special case of Theorem 4.2.6.

4.2.4 An Example: Diffie-Hellman Key Exchange

The general setting. Consider the following simple example: two honest parties, e.g., Alice and Bob, perform a Diffie-Hellman key exchange using authenticated communication and then extract an actual key by hashing the group element g^{ab}, while an eavesdropper is present.

Since both the honest parties hash the exactly same element, there is no necessity to treat them as different parties and we can work in the indifferentiability setting with one honest party and the adversary. Consider the following resources: let DH be a Diffie-Hellman resource (modeling the authenticated key exchange) that outputs g^{ab} at interface $\mathtt{A}$ and (g^a, g^b) at interface $\mathtt{E}$, let PO denote a random oracle accessible by both parties, let HK denote a public hash key resource that outputs the key at both interfaces, and let KEY be a resource that outputs a uniformly random key at interface $\mathtt{A}$ and nothing at interface $\mathtt{E}$. The Diffie-Hellman converter π takes the group element g^{ab} at the inside interface, inputs it to the random oracle, and outputs the obtained result at the outside interface. It is easy to see that under the CDH assumption we have $[\mathsf{PO}, \mathsf{DH}] \xrightarrow[\text{asympt}]{\pi} \mathsf{KEY}$, using the simulator σ that chooses (g^a, g^b) uniformly at random and simulates Eve's interface of the public random oracle.

Limitation of indifferentiability. The explicit appearance of the resource PO in the above statement corresponds to a proof in the so called random oracle model. The corresponding simulator σ chooses (g^a, g^b) uniformly at random and simulates the interface $\mathtt{E}$ of the public random oracle[1]. If we want to obtain a proof in the standard model, i.e., getting rid of the assumed random oracle resource, we would need to find a (potentially) keyed hash function that instantiates the random oracle, which is of course impossible. Such a hash function is in our terminology just a converter h that reduces the random oracle to the public hash key resource HK, i.e., $\mathsf{HK} \xrightarrow[\text{asympt}]{\boldsymbol{h}} \mathsf{PO}$. If we had such a hash function, we could use parallel composition to obtain $[\mathsf{HK}, \mathsf{DH}] \xrightarrow[\text{asympt}]{\boldsymbol{h}} [\mathsf{PO}, \mathsf{DH}]$ and then sequential composition to obtain $[\mathsf{HK}, \mathsf{DH}] \xrightarrow[\text{asympt}]{\boldsymbol{\pi} \circ \boldsymbol{h}} \mathsf{KEY}$.

Using context-restricted constructions. The main obstacle in the way of the modular approach is that there exists no hash function h that constructs the random oracle from a public hash key. However, it might be feasible within an appropriate context set $\mathcal{C}$. Composing this with the second step should then be possible as long as the protocol which we want to actually apply is in the context set, i.e., $(\boldsymbol{\pi}, \mathsf{DH}) \in \mathcal{C}$. We now show, that this is exactly what the composition theorem of context-restricted constructions yields:

Assume that $\mathsf{HK} \xrightarrow[\text{cr-asym}]{\boldsymbol{h}, \mathcal{C}} \mathsf{PO}$ for some context set $\mathcal{C}$ with $(\boldsymbol{\pi}, \mathsf{DH}) \in \mathcal{C}$. Let $\mathcal{C}' \coloneqq \{(\boldsymbol{\pi}, \square)\}$. According to the parallel composition rule of Theorem 4.2.6, we have that $[\mathsf{HK}, \mathsf{DH}] \xrightarrow[\text{cr-asym}]{\boldsymbol{h}, \mathcal{C}'} [\mathsf{PO}, \mathsf{DH}]$, since by definition of the the neutral resource, $(\boldsymbol{\pi}, [\mathsf{DH}, \square])$ is equivalent to $(\boldsymbol{\pi}, \mathsf{DH})$ and thus contained in $\mathcal{C}$. Using Proposition 4.2.5, we moreover have $[\mathsf{PO}, \mathsf{DH}] \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi}, \mathcal{C}_{id}} \mathsf{KEY}$, and since by definition $(\boldsymbol{id} \circ \boldsymbol{\pi}, \square) = (\boldsymbol{\pi}, \square) \in \mathcal{C}'$, we can apply sequential composition and obtain $[\mathsf{HK}, \mathsf{DH}] \xrightarrow[\text{cr-asym}]{\boldsymbol{\pi} \circ \boldsymbol{h}, \mathcal{C}_{id}} \mathsf{KEY}$, which is equivalent to $[\mathsf{HK}, \mathsf{DH}] \xrightarrow[\text{asympt}]{\boldsymbol{\pi} \circ \boldsymbol{h}} \mathsf{KEY}$.

[1] The fact that the random oracle "vanishes" and is simulated in the ideal world corresponds to the notion of a programmable random oracle.

In summary, this shows that the composition theorem of the context-restricted construction notion yields exactly what one expects: composition works if and only if the considered application is in the set of allowed contexts. In fact, the type of contexts for which this construction works, is called split security in the UCE framework. Split security is discussed in more detail in Section 4.5.

4.3 UCE as a Special Case

In the following section, we consider context-restricted constructions of random oracle resources in the so-called indifferentiability setting. Recall, that indifferentiability considers outbound resources with two parties: one honest and one dishonest. We then prove that the UCE framework is actually a special case of such constructions.

4.3.1 Constructing Random Oracles

In the following, let $H\colon H.\mathcal{K}\times H.\mathcal{X} \to H.\mathcal{Y}$ denote a keyed hash function, let HK_H denote the public hash-key resource that chooses a key for H and outputs it at both interfaces, let hash_H denote the converter that implements an oracle for H at the outside interface when connected to HK_H at the inside interface, and let $\mathsf{H} \coloneqq \mathbf{hash}_H\mathsf{HK}_\mathsf{H}$ as a shorthand. Finally, let RO_H denote the private random oracle resource with the same input and output domains as H, where by private we mean that this resource only accepts queries at interface $\mathtt{A}$.[2] See Figure 4.4 for a formal description of these resources and converters.

We now present an alternative formalization of UCE based on context-restricted constructions, more concretely that every possible UCE-class $\mathcal{S}^x$, where $x \in \{\mathrm{sup}, \mathrm{cup}, \mathrm{srs}, \mathrm{crs}, \mathrm{splt}, \ldots\}$, can be mapped to a set of contexts $\mathcal{C}^x$ for which the UCE statement implies the context-restricted construction statement $\mathsf{HK}_\mathsf{H} \vdash \xrightarrow[\text{cr-asym}]{\mathbf{hash}_H,\mathcal{C}} \mathsf{RO}$, and moreover, if the construction statement were restricted to a specific simulator, the reverse direction would hold as well.

[2] The choice to consider a private random oracle stems from the fact that in the UCE framework the hash key is just chosen uniformly at random instead of allowing an arbitrary efficient simulator with access to the random oracle to generate this key.

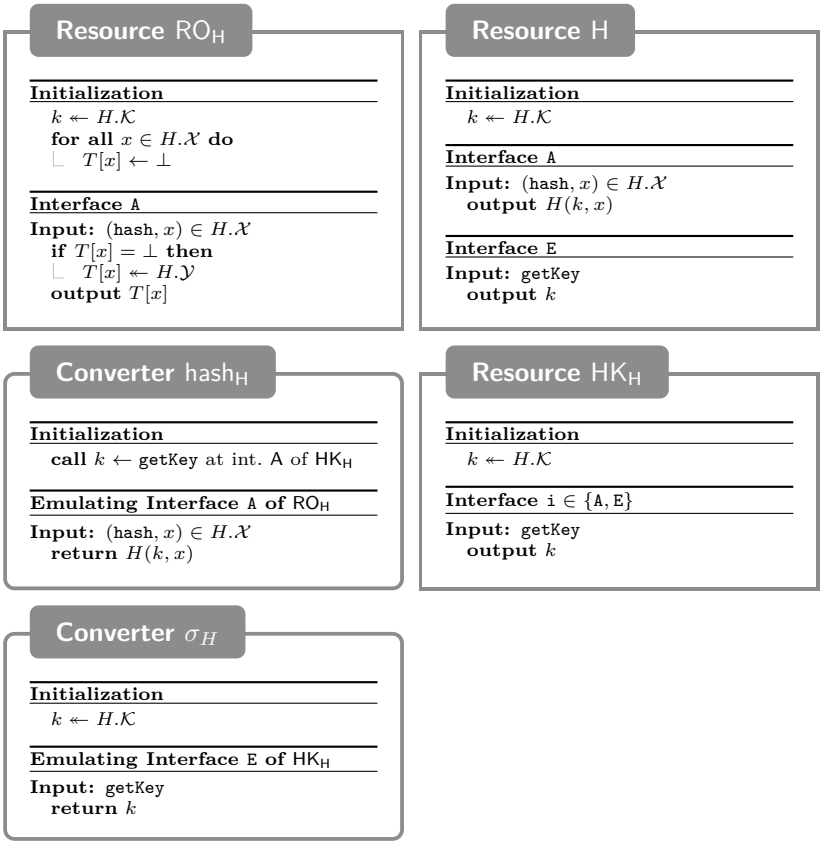


Figure 4.4: Formal definitions of the random oracle resource, the corresponding hash converter and hash-key resource, the shorthand resource $\mathsf{H} \coloneqq \mathbf{hash}_H \mathsf{HK}_\mathsf{H}$, and the simulator hard-coded by the UCE game.

4.3.2 Non-Interactive Contexts

In order to map every UCE-class to an equivalent set of contexts, we first introduce the set of non-interactive contexts, i.e., the communication between the source and the distinguisher being unidirectional. This restricted set of contexts faithfully encodes the structural restrictions of the traditional UCE game (cf. page 42), where the communication between the source and the distinguisher is unidirectional. Recall that we are in the same general setting as the classical indifferentiability framework, where one only considers out-bound resources for which communication at one interface does not affect the other interface.

Definition 4.3.1. A *non-interactive resource* P is a resource that at the interface `E` accepts at most a single trigger query (usually called `retrieve`), and a *non-interactive filter* is a converter that at the outer interface just accepts a single trigger query (usually called `retrieve`). Let Θ^{ni} denote the set of all non-interactive resources, and Σ^{ni} denote the set of all non-interactive filters, respectively.

Each UCE-source naturally corresponds to a set of non-interactive contexts. This is formally stated in the following lemma by providing a surjective mapping from the set of non-interactive contexts to the set of UCE sources $\mathcal{S}$.

Lemma 4.3.2. *The function* $\phi\colon \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}} \to \mathcal{S}$ *that maps every context* $(\mathbf{f}, \mathsf{P})$ *to the following UCE source* S*, that internally emulates* $\mathbf{f}$ *and* P*, is surjective.*

1. *S queries the interface* `E` *of* P *to obtain z.*
2. *S queries the outside interface of the filter* $\mathbf{f}$ *to obtain y. The queries at the inside interface of* $\mathbf{f}$ *are forwarded to the resource* P *or output as queries to the hash oracle, respectively.*
3. *S outputs $L = (y, z)$.*

Proof. First, it is easy to see that ϕ is indeed a function from $\Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}}$ to $\mathcal{S}$, i.e., $\phi(\mathbf{f}, \mathsf{P})$ is a valid UCE source for every context $(\mathbf{f}, \mathsf{P}) \in \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}}$. To see that this function is surjective, fix an arbitrary source S. Now, let $\mathbf{f}_S$ denote the filter that upon receiving the query `retrieve` at the outer interface internally runs S and answers this query with the leakage L.

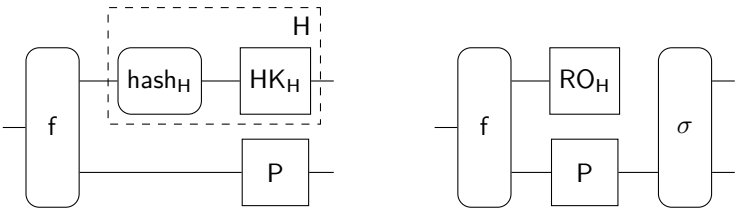


Figure 4.5: The real (left) and the ideal (right) setting of context-restricted indifferentiability when applied to UCE.

Each hash query of S is forward towards the attached random oracle or hash resource, and the corresponding answer is forwarded to S. Clearly $\phi(\mathbf{f}_S, \square) = S$, where $\square \in \Theta^{\mathsf{ni}}$ denotes the dummy resource. □

4.3.3 RO-CRI Security Implies UCE Security

We now show, that for the specific simulator $\boldsymbol{\sigma}_H$ that chooses the hash key uniformly at random, the distinguishing problem of a context-restricted indifferentiability statement, for a fixed context $(\mathbf{f}, \mathsf{P})$, on random oracles is as hard as the UCE game with the source $\phi(\mathbf{f}, \mathsf{P})$. In order to relate more directly to the traditional UCE definition, we first introduce the RO-CRI advantage, which is depicted in Figure 4.5 for a specific context $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}$.

Definition 4.3.3. We define the *random-oracle context-restricted indifferentiability (RO-CRI)* advantage of a distinguisher D on a hash function H in a context $(\mathbf{f}, \mathsf{P})$ as

$$\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}}(\mathsf{D}) \coloneqq \Delta^{\mathsf{D}}(\mathbf{f}[\mathsf{H},\mathsf{P}], \mathbf{f}[\mathsf{RO}_H,\mathsf{P}]\boldsymbol{\sigma}),$$

for a simulator $\boldsymbol{\sigma}$. If there exists a simulator $\boldsymbol{\sigma}$ such that for all efficient distinguishers and all contexts $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}$, the RO-CRI advantage is negligible, we say that H is $\mathcal{C}$ random-oracle context-restricted indifferentiable.

The following lemma implies that for non-interactive contexts this definition is equivalent to the game-based definition of UCE-security, if we fix the simulator to $\boldsymbol{\sigma}_H$.

Lemma 4.3.4. *Let $\mathcal{S}$ denote the set of all UCE-sources and $\phi\colon \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}} \to \mathcal{S}$ the surjective function from Lemma 4.3.2. For every distinguisher D, there is a distinguisher D' (with essentially the same efficiency) with*

$$\forall(\mathbf{f},\mathsf{P}) \in \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}}\colon \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_H}(\mathsf{D}) = \mathbf{Adv}^{\text{uce}}_{H,\phi(\mathbf{f},\mathsf{P}),\mathsf{D}'},$$

where $\mathbf{Adv}^{\text{uce}}_{H,S,D}$ denotes the uce-advantage of (S,D) on H. Conversely, for every distinguisher D' there is an equally efficient distinguisher D such that for all $(\mathbf{f},\mathsf{P}) \in \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}}$ we have $\mathbf{Adv}^{\text{uce}}_{H,\phi(\mathbf{f},\mathsf{P}),\mathsf{D}'} = \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_H}(\mathsf{D})$.

Proof. For every distinguisher D for $\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_H}(\mathsf{D})$ we can construct a distinguisher D' using a wrapper around D as follows: if D queries the interface $\mathtt{E}$ of the hash resource (for the key) or P we return hk or z, respectively; if D queries the outer interface of $\mathbf{f}$, then y is returned. The bit b' is then set to the output bit of D. The key observation is that the resources $\mathbf{f}[\mathsf{H},\mathsf{P}]$ and $\mathbf{f}[\mathsf{RO},\mathsf{P}]\boldsymbol{\sigma}_H$ are independent to the order in which D does those queries. It is now easy to see that $\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_H}(\mathsf{D}) = \mathbf{Adv}^{\text{uce}}_{H,\phi(\mathbf{f},\mathsf{P}),\mathsf{D}'}$.

The reverse direction works with an analogous wrapper that first queries the system to obtain hk, z, and y. It then invokes D' with hk and $L = (y,z)$ as inputs and outputs the bit b'. □

We now state the main result of this section, relating the UCE game to context-restricted indifferentiability. It implies that instead of viewing the source as the first stage of an adversary, one can view it as the set of contexts in which the hash function can safely be used.

Theorem 4.3.5. *Let $\mathcal{D}$ denote the set of all efficient distinguishers. For every class $\mathcal{S}^x$ of UCE sources, there exists a set of contexts $\mathcal{C}^x$ such that $\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_H}(\mathsf{D})$ is negligible for every $D \in \mathcal{D}$ and every context $(\mathbf{f},\mathsf{P}) \in \mathcal{C}^x$ if and only if $\mathbf{Adv}^{\text{uce}}_{H,S,D}(\cdot)$ is negligible for all $(S,D) \in \mathcal{S}^x \times \mathcal{D}$.*

Proof. Using the surjectivity of ϕ (Lemma 4.3.2), we have that for any UCE-class $\mathcal{S}^x$ we can define $\mathcal{C}^x := \phi^{-1}(\mathcal{S}^x)$ such that $\phi(\mathcal{C}^x) = \mathcal{S}^x$. Hence, by Lemma 4.3.4 we have that $\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_H}(\mathsf{D})$ is negligible for all efficient distinguishers $\mathsf{D} \in \mathcal{D}$ and all contexts $(\mathbf{f},\mathsf{P}) \in \mathcal{C}^x$ iff $\mathbf{Adv}^{\text{uce}}_{H,S,D}(\cdot)$ is negligible for all $(S,D) \in \mathcal{S}^x \times \mathcal{D}$. □

The following corollary establishes the unidirectional implication from UCE-security to context-restricted indifferentiability. The reverse direction does not necessarily hold, since the context-restricted indifferentiability notion allows for different simulators than the natural one $\boldsymbol{\sigma}_H$.

Corollary 4.3.6. *Let $\mathcal{D}$ denote the set of all efficient distinguishers. For every class $\mathcal{S}^x$ of UCE sources, there exists a set of contexts $\mathcal{C}^x$ such that if $\mathbf{Adv}^{\text{uce}}_{H,S,D}(\cdot)$ is negligible for all $(S, D) \in \mathcal{S}^x \times \mathcal{D}$, then $\mathsf{HK}_\mathsf{H} \vdash \xrightarrow[\text{cr-asym}]{\mathbf{hash}_H, \mathcal{C}^x} \mathsf{RO}_\mathsf{H}$.*

Proof. This follows directly from Definitions 4.2.3 and 4.3.3 and Theorem 4.3.5. □

4.4 Public-Seed PRPs as a Special Case

In [ST17] Soni and Tessaro introduce the notion of public-seed pseudorandom permutations (psPRP) that are inspired by UCE. In fact, they introduce a generalization of UCE, called public-seed pseudorandomness (psPR), of which both psPRP and UCE are instantiations. In the following, we give an analogous equivalence result to the one of Section 4.3 between context-restricted indifferentiability and the general public-seed pseudorandomness notion. The equivalence for the psPRP notion then just follows as a trivial corollary.

4.4.1 Public-Seed Pseudorandomness

We first briefly recap the main definitions of public-seed pseudorandomness as introduced in [ST17]. The authors first introduce the notion of an ideal primitive, of which both random oracles and ideal random permutations are instantiations of.

Definition 4.4.1. An ideal primitive is a pair $I = (\Sigma, D)$, where $\Sigma = \{\Sigma_\lambda\}_{\lambda\in\mathbb{N}}$ is a family of sets of functions (such that all functions have the same domain and range), and $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$is a family of probability distributions, where the range of D_λ is a subset of Σ_λ for all $\lambda \in \mathbb{N}$. The ideal primitive I, once the security parameter λ is fixed, should be thought of as an oracle that initially samples a function I as

its initial state according to D_λ from Σ_λ. Then, I provides access to I via queries i.e. on input x it returns $I(x)$.

Moreover, the authors of [ST17] then define the following notion of Σ-compatible function families. A function family corresponds to an algorithm that generalizes hash functions and pseudo-random permutations.

Definition 4.4.2. A function family $\mathrm{F} = (\mathrm{Kg}, \mathrm{Eval})$ consists of a key (or seed) generation algorithm F.Kg and an evaluation algorithm F.Eval.

- F.Kg is a randomized algorithm that on input the unary representation of the security parameter λ returns a key k, and we let $[\mathrm{F.Kg}(1^\lambda)]$ denote the set of all possible outputs of $\mathrm{F.Kg}(1^\lambda)$.
- F.Eval is a deterministic algorithm that takes three inputs; the security parameter in unary form 1^λ, a key $k \in [\mathrm{F.Kg}(1^\lambda)]$ and a query x such that $\mathrm{F.Eval}(1^\lambda, k, \cdot)$ implements a function that maps queries x to $\mathrm{F.Eval}(1^\lambda, k, x)$.

We say that F is efficient if both Kg and Eval are polynomial-time algorithms.

The goal of such a function family F is then to implement an ideal primitive I with respect to the UCE-like security game depicted in Figure 4.6, considering an adversary that is split into a source S and a distinguisher D. In contrast to the original definition, we only consider the game for a single session, which can easily be related to the multi-session one using a standard hybrid argument. Finally, Soni and Tessaro define the pspr-advantage as follows:

$$\mathbf{Adv}^{\mathsf{pspr}[I]}_{\mathrm{F},S,D}(\lambda) = 2\Pr[\mathsf{psPR}^{S,D}_{\mathrm{F},I}(\lambda) = 1] - 1.$$

4.4.2 Ideal Primitives and Function Families

In the following section, we argue that every ideal primitive I can be understood as an ideal resource of an context-restricted indifferentiability statement, and every function family F as an pair of real resource and protocol, respectively. For simplicity, we ignore the security parameter λ in the following.

The psPR game

function Main $\mathrm{psPR}^{S,D}_{\mathrm{F},I}(\lambda)$
  $b \leftarrow \{0,1\}$
  $k \leftarrow \mathrm{F.Kg}(1^\lambda)$
  $f \leftarrow I_\lambda$
  $L \leftarrow S^{\mathcal{O}}(1^\lambda)$
  $b' \leftarrow D(1^\lambda, k, L)$
  return $(b' = b)$

function $\mathcal{O}(x)$
  if $b = 1$ **then**
    return $\mathrm{F.Eval}(1^\lambda, k, x)$
  else
    return $f(x)$

Figure 4.6: The public-seed pseudorandomness security game for a function family F, an ideal primitive I, a source S, and a distinguisher D.

For every ideal primitive I and for every function family $\mathrm{F} = (\mathrm{Kg}, \mathrm{Eval})$, denote the corresponding resource and converters depicted in Figure 4.7. Moreover, we also define the simulator $\boldsymbol{\sigma}_F$, which simply chooses a key according to Kg as well.

4.4.3 CRI-Security Implies psPR-Security

We now show, that for the specific simulator $\boldsymbol{\sigma}_{KG}$, if for every specific context $(\mathbf{f}, \mathsf{P})$ the distinguishing problem of context-restricted indifferentiability (CRI) is hard, then the UCE game with the fixed source $\phi(\mathbf{f}, \mathsf{P})$ is hard as well, and vice versa. In order to relate more directly, we introduce the psRP-CRI advantage.

Definition 4.4.3. We define the *public-seed pseudorandomness context-restricted indifferentiability (psRP-CRI)* advantage of a distinguisher D on a hash function H in a context $(\mathbf{f}, \mathsf{P})$ as

$$\mathbf{Adv}^{\mathrm{psPR-CRI}}_{\mathrm{F},\mathsf{I},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) \coloneqq \Delta^{\mathsf{D}}(\mathbf{f}[\mathbf{eval}_F \mathsf{KG_F}, \mathsf{P}], \mathbf{f}[\mathsf{I}, \mathsf{P}]\boldsymbol{\sigma}),$$

for a simulator $\boldsymbol{\sigma}$.

The following lemma implies that for non-interactive contexts this definition is equivalent to the game-based definition of UCE security, if we fix the simulator to $\boldsymbol{\sigma}_F$.

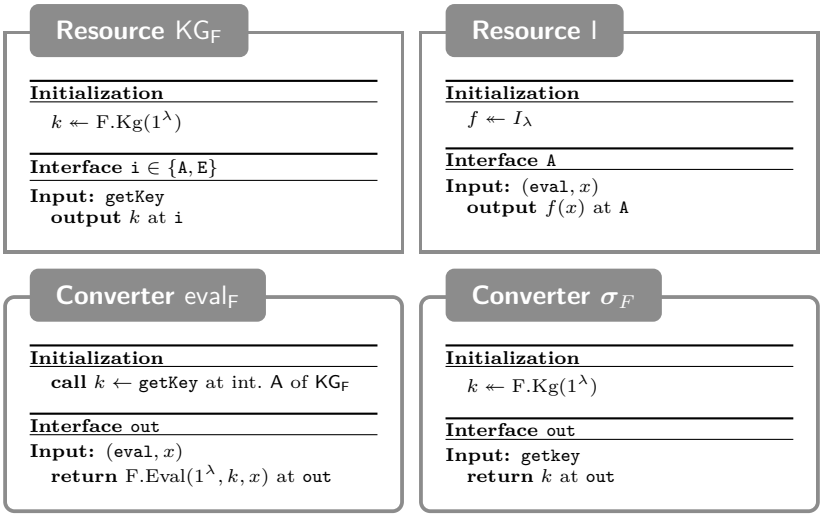


Figure 4.7: The resources and converters corresponding to the public-seed pseudorandomness notion.

Lemma 4.4.4. *Let* F *denote a function family and* I *an ideal primitive. Furthermore, let* $\mathcal{S}$ *denote the set of all psPR-sources and* $\phi\colon \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}} \to \mathcal{S}$ *the surjective function from Lemma 4.3.2. For every distinguisher* D*, there is a distinguisher* D' *(with essentially the same efficiency) with*

$$\forall(\mathbf{f},\mathsf{P}) \in \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}}\colon \mathbf{Adv}^{\mathrm{psPR-CRI}}_{\mathrm{F},\mathsf{I},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_F}(\mathsf{D}) = \mathbf{Adv}^{\mathsf{pspr}[I]}_{\mathrm{F},\phi(\mathbf{f},\mathsf{P}),\mathsf{D}'}$$

Conversely, for every distinguisher D' *there is a distinguisher* D *(with essentially the same efficiency) such that for all* $(\mathbf{f},\mathsf{P}) \in \Sigma^{\mathsf{ni}} \times \Theta^{\mathsf{ni}}$ *we have* $\mathbf{Adv}^{\mathsf{pspr}[I]}_{\mathrm{F},\phi(\mathbf{f},\mathsf{P}),\mathsf{D}'} = \mathbf{Adv}^{\mathrm{psPR-CRI}}_{\mathrm{F},\mathsf{I},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_F}(\mathsf{D})$.

Proof. The proof is analogous to the one of Lemma 4.3.4.

We can now state the main result of this section, relating the public-seed pseudorandomness game to context-restricted indifferentiability.

Theorem 4.4.5. *Let* F *denote a function family,* I *an ideal primitive, and let* $\mathcal{D}$ *denote the set of all efficient distinguishers. For*

every family $\mathcal{S}^x$ of psPR-sources, there exists a set of contexts $\mathcal{C}^x$ such that $\mathbf{Adv}^{\mathrm{psPR-CRI}}_{\mathrm{F},\mathsf{I},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_F}(\mathsf{D})$ is negligible for every $D \in \mathcal{D}$ and every context $(\mathbf{f},\mathsf{P}) \in \mathcal{C}^x$ if and only if $\mathbf{Adv}^{\mathsf{pspr}[I]}_{\mathrm{F},S,D}(\cdot)$ is negligible for all $(S,D) \in \mathcal{S}^x \times \mathcal{D}$.

Proof. The proof is analogous to the one of Theorem 4.3.5.

This demonstrates that not only UCE is a special case of CRI but also the more general notion of psPR is still a special case of CRI, where each ideal primitive and function family correspond to the ideal and real world, respectively. Similarly to UCE, the psPR notion is still non-interactive and essentially hard-codes a specific simulator in the security game.

4.5 Generalizing Split-Security

In this section, we present generalizations of the split-source UCE-class, that cannot be formalized in plain UCE, based on context-restricted indifferentiability.

4.5.1 Split-Security

The split-source UCE-class has been proposed by Bellare et al. after it has been shown that computational-unpredictable UCE-security and computational-reset-secure UCE-security is infeasible if indistinguishability obfuscation exists. Note that split-security is not a stand-alone UCE-class in the sense that it is designed to be combined with either computational unpredictability or reset-security, respectively.

The general idea of split-security is, that the source must not be able to compute $\mathrm{iO}(H(\cdot, x) = y)$, i.e., an obfuscation of a circuit validating whether a given hash key was used to evaluate the hash function. To achieve this, the source must be dividable into two parts (S_0, S_1), where S_0 chooses a vector $(x_1, \ldots, x_n)$ of query points, without having access to the hash oracle, and S_1 then just gets the evaluations $y_i \coloneqq \mathrm{Hash}(x_i)$, without having access to the hash oracle either. Thus, no part of the source knows both x_i and its evaluation y_i, preventing the aforementioned iO attack. A formal description of the split-source $S \coloneqq \mathit{Splt}[S_0, S_1]$ is found in Figure 4.8.

Splt SOURCE

function $\text{SPLT SOURCE}^{\text{HASH}}(1^\lambda)$
  $(L_0, x) \twoheadleftarrow S_0(1^\lambda)$
  for $i = 1, \ldots, |x|$ **do**
    $y[i] \leftarrow \text{HASH}(x[i])$
  $L_1 \twoheadleftarrow S_1(1^\lambda, y)$
  $L \leftarrow (L_0, L_1)$
  return L

Figure 4.8: The definition of the split-source family in UCE.

4.5.2 An Alternative Representation

As established by Theorem 4.3.5, using $\mathcal{C}^{\mathsf{splt}} := \phi^{-1}(\mathcal{S}^{\mathsf{splt}})$ faithfully translates split-security to context-restricted indifferentiability. In order to work more easily with split-security and make it more directly comparable to our later generalizations thereof, however, we introduce an alternative representation of the split-security RO-CRI context set using a fixed filter $\mathbf{f}^{\mathsf{splt}}$, which encodes the structural restriction of split-security.

Definition 4.5.1. The *split* RO-CRI context set is the set of filters and non-interactive resource pairs of which the filter can be factorized into the filter $\mathbf{f}^{\mathsf{splt}}$, as depicted in Figure 4.9, followed by an arbitrary filter. Formally,

$$\mathcal{C}^{\mathsf{splt}} := \{\mathbf{f} \circ \mathbf{f}^{\mathsf{splt}} \mid \mathbf{f} \in \Sigma^{\mathsf{ni}}\} \times \Theta^{\mathsf{ni}}.$$

Observe that the filter $\mathbf{f}^{\mathsf{splt}}$ expects the resource P to output a sequence of pairs (x_i, a_i), where x_i is intended to be unpredictable, then hashes $x_i \parallel a_i$ and outputs the result. Note that the distinction into an unpredictable value x_i and an auxiliary value a_i solely prepares for our generalizations. This type of resource corresponds to the first stage of the source S_0 that produces the queries[3] and the leakage L_0 (called Z in the following definition), and we will call it a *seed* in the following.

[3]Here, we only consider split sources with a fixed number of queries. A polynomial number of queries could easily be phrased as well.

Definition 4.5.2. A *seed with n outputs* is a resource that initially draws random values $X_1, \ldots, X_n$, $A_1, \ldots, A_n$, and Z according to some joint distribution. Then, it accepts at the interface $\mathtt{E}$ a single trigger query (usually called `retrieve`) that is answered with Z, and at the interface $\mathtt{A}$ n trigger queries answered with (X_1, A_1) to (X_n, A_n). Let $\Theta_n^{\mathsf{seed}} \subset \Theta^{\mathsf{ni}}$ denote the set of all seeds with n outputs. Moreover, let $\mathcal{C}_n^{\mathsf{seed}} := \Sigma \times \Theta_n^{\mathsf{seed}}$.

The second stage of the source S_1 then translates to the additional non-interactive filter $\mathbf{f}$ that gets from $\mathbf{f}^{\mathsf{splt}}$ the hashed values y_i and can further process them to obtain the leakage L_1. The following lemma establishes that this represents a correct translation of split-security as well.

Lemma 4.5.3. *Let $\mathcal{S}^{\mathsf{n}}$ denote the class of all UCE sources making at most n oracle queries and let ϕ denote the surjective function from Lemma 4.3.2. We then have*

$$\phi\big(\mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}_n^{\mathsf{seed}}\big) = \mathcal{S}^{\mathsf{splt}} \cap \mathcal{S}^{\mathsf{n}},$$

and thus, $\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\boldsymbol{\sigma}_H}(\mathsf{D})$ is negligible for every $D \in \mathcal{D}$ and every context $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}_n^{\mathsf{seed}}$ if and only if $\mathbf{Adv}^{\text{uce}}_{H,S,D}(\cdot)$ is negligible for all $(S, D) \in (\mathcal{S}^{\mathsf{splt}} \cap \mathcal{S}^{\mathsf{n}}) \times \mathcal{D}$.

Proof (Sketch). First, we show that for every $\mathbf{f} \in \Sigma^{\mathsf{ni}}$ and $\mathsf{P} \in \Theta_n^{\mathsf{seed}}$ the context $(\mathbf{f} \circ \mathbf{f}^{\mathsf{splt}}, \mathsf{P})$ is mapped to a UCE source in $\mathcal{S}^{\mathsf{splt}} \cap \mathcal{S}^{\mathsf{n}}$ by ϕ. To this end, we define S_0 to be the source which initially emulates P. It first queries z at the interface $\mathtt{E}$ and all values $x = x_1, \ldots, x_n$ at the interface $\mathtt{A}$ of P and set $L_0 = z$. The source S_1 internally emulates $\mathbf{f}$. It initially queries `retrieve` towards $\mathbf{f}$ to obtain L_1. Whenever $\mathbf{f}$ outputs a query `retrieve` towards $\mathbf{f}^{\mathsf{splt}}$, then S_1 answers by using the next value y_i. Now observe that, by definition of Θ^{ni}, obtaining all queries $x_1, \ldots, x_n$ from P at interface $\mathtt{A}$ on demand or at the beginning and storing the results $y_1 = H(k, x_1), \ldots, y_n = H(k, x_n)$ is equivalent. Thus, it is easy to verify that $\phi(\mathbf{f} \circ \mathbf{f}^{\mathsf{splt}}, \mathsf{P}) = \mathit{Splt}[S_0, S_1]$.

Second, we show that $\phi(\mathcal{C}_n^{\mathsf{splt}}) \supseteq \mathcal{S}^{\mathsf{splt}} \cap \mathcal{S}^{\mathsf{n}}$, i.e., for every split source there exists at least one context that maps to this source. It is easy to see that S_0 can be embedded accordingly in a resource $\mathsf{P} \in \Theta_n^{\mathsf{seed}}$ and S_1 in a filter $\mathbf{f} \in \Sigma^{\mathsf{ni}}$ such that $\phi(\mathbf{f} \circ \mathbf{f}^{\mathsf{splt}}, \mathsf{P}) = \mathit{Splt}[S_0, S_1]$. The remaining claim

Converter $\mathrm{f}^{\mathsf{splt}}$

Outer Interface

Input: `retrieve`
  call $v \leftarrow$ `retrieve` at int. A of P
  if v can be parsed as $(x, a) \in \mathcal{X} \times \mathcal{A}$ **then**
    call $y \leftarrow (\mathtt{hash}, (x \parallel a))$ at int. A of RO_{H}
    return y
  else
    return $\perp$ at `out`

Figure 4.9: The definition of the filter $\mathbf{f}^{\mathsf{splt}}$. The filter is implicitly parametrized in the two sets $\mathcal{X}$ and $\mathcal{A}$, which should become clear from the context in all of our uses.

about the advantages immediately follows by Lemma 4.3.4, concluding the proof. □

4.5.3 Strong-Split Security

Split sources have several limitations. First, the distinguisher cannot influence the queries at all and, thus, all queries must be solely determined by the honest parties. This prevents, for example, queries like $H(hk, x \| a)$ where a is a value which can be chosen by the distinguisher (e.g. a is transmitted over an insecure channel) even if x is unpredictable. In the following section, we introduce a generalization of split-security, called *strong-split* security, to address this limitation. Second, split-security does not allow nested queries like $H(hk, H(hk, x))$. In Section 4.5.5 we present a further generalization to address this issue as well.

Remark. Note that the first limitation is not specific to split-security, but is inherent to the traditional UCE-game. In their work [FM16] on Interactive Computational Extractors (ICEs), Farshim and Mittelbach have proposed an alternative relaxation of this issue. In Section 4.5.6 we show that ICE security implies strong-split context-restricted indifferentiability for statistical unpredictability.

Converter $\mathbf{f}_p^{\mathsf{s\text{-}splt}}$

Outer Interface

Input: $(\texttt{retrieve}, f_1, \ldots f_p) \in \mathcal{I}^p_{\mathcal{X}\times\mathcal{A}\to H.\mathcal{X}}$
  call $v \leftarrow \texttt{retrieve}$ at int. A of P
  if v can be parsed as $(x, a) \in \mathcal{X} \times \mathcal{A}$ **then**
    for $i = 1, \ldots, p$ **do**
      $y[i] \leftarrow f_i(x, a)$
    if $\forall i \neq j \colon y[i] \neq y[j]$ **then**
      for $i = 1, \ldots, p$ **do**
        call $z[i] \leftarrow (\texttt{hash}, y[i])$ at int. A of RO_H
  if z is not set **then** $z \leftarrow \bot^p$
  return z

Figure 4.10: The strong-split filter $\mathbf{f}_p^{\mathsf{s\text{-}splt}}$ for RO-CRI, where $\mathcal{I}_{\mathcal{X}\times\mathcal{A}\to H.\mathcal{X}}$ denotes the set of all efficiently computable functions from $\mathcal{X}\times\mathcal{A}$ to $H.\mathcal{X}$ that are injective in the first argument. Note that it was pointed out in [BM14] that the queries of a split-source must be distinct; otherwise arbitrary information can be communicated to the second stage.

In order to allow the distinguisher to influence the queries while ensuring that the overall query is still unpredictable from the viewpoint of the distinguisher, we allow him to apply any *injective* function on the preliminary inputs x specified by the first part of the source S_0, which will then be evaluated and passed on to S_1. That is, we use the simple fact that for any injective function f guessing $f(x_i)$ is at least as hard as guessing x_i. To formally model this as a context set for RO-CRI, we use a specific filter $\mathbf{f}_p^{\mathsf{s\text{-}splt}}$. This filter expects the resource P to output a sequence of pairs (x_i, a_i), where x_i is intended to be unpredictable. We will call such a resource P *seed* in the following. For each of them the distinguisher can then input p functions $f_i^1, \ldots, f_i^p$ that are injective in the first arguments, upon which the filter will output $(f_i^1(x_i, a_i), \ldots, f_i^p(x_i, a_i))$ to the hash oracle and forwards the results to the distinguisher. A formal definition of is depicted in Figure 4.10. The filter $\mathbf{f}_p^{\mathsf{s\text{-}splt}}$ can then be combined with an arbitrary non-interactive resource to obtain a strong-split RO-CRI context.

Definition 4.5.4. The *strong-split* RO-CRI context set is the set of filters and non-interactive resource pairs of which the filter can be

factorized into $\mathbf{f}_p^{\mathsf{s\text{-}splt}}$ followed by an arbitrary filter. Formally,

$$\mathcal{C}_p^{\mathsf{s\text{-}splt}} \coloneqq \{\mathbf{f} \circ \mathbf{f}_p^{\mathsf{s\text{-}splt}} \mid \mathbf{f} \in \Sigma\} \times \Theta^{\mathsf{ni}}.$$

Analogous to split-security, strong-split security is not a sufficient restriction to avoid trivial impossibility results. Rather, these notions are meant to be combined with a notion of unpredictability or reset-security. However, for strong-split security, requiring the seed to output distinct unpredictable values is still insufficient to guarantee the security: for instance, if the resource P outputs (x, a_1) and $(x+1, a_2)$, then the distinguisher can easily choose f and g such that $f(x, a_1) = g(x+1, a_2)$. Therefore, we introduce suitable notion of unpredictability in the next subsection, which when combined with strong-split security presents a plausible assumption for a hash function family.

4.5.4 Strict Min-Entropy Seeds

We now define an information-theoretic restriction on the seed called *strict min-entropy seeds*. Similar to Farshim and Mittelbach [FM16] we choose to focus on statistical rather than computational unpredictability to ensure that our notion excludes interactive version of the attack highlighted in [BFM14].[4] More concretely, we consider seeds whose outputs at interface $\mathtt{A}$ consist of pairs (X_i, A_i), with A_i being an auxiliary value, such that X_i has high *average conditional min-entropy* given the leakage Z and all previous queries.

Definition 4.5.5. A *strict min-entropy k-bit seed with n outputs* is seed with n outputs (cf. Definition 4.5.2), such that

$$\forall i \leq n\colon\ \tilde{H}_\infty\big(X_i \,\big|\, \{X_j\}_{j<i}, \{A_j\}_{j\leq i}, Z\big) \ \geq\ k.$$

Let $\Theta_{n,k}^{\mathsf{s-me}} \subset \Theta^{\mathsf{ni}}$ denote the set of all strict min-entropy k-bit seed with n outputs. Moreover, let $\mathcal{C}_{n,k}^{\mathsf{s-me}} \coloneqq \Sigma \times \Theta_{n,k}^{\mathsf{s-me}}$ denote the set of all strict min-entropy k-bit contexts.

When combining split-security or strong-split security with strict min-entropy seeds, the security does not depend on the maximal number n of values produced by the seed.

[4] We would like to stress that while split-security was originally introduced for the computational setting, it is still a natural class to consider even when combined with a statistical unpredictability notion.

Lemma 4.5.6. *Let n be polynomially bounded. If H is a $\mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}^{\mathsf{s-me}}_{1,k}$ RO-CRI secure, then H is also $\mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}$ RO-CRI secure.*

More concretely, let $\mathcal{D}$ denote the set of distinguishers. Then there exists $\rho\colon \mathcal{D} \times \left(\mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}\right) \to \mathcal{D}$ and $\psi\colon \mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k} \to \mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}^{\mathsf{s-me}}_{1,k}$, such that for every $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}$ we have

$$\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) \leq \binom{n}{2} 2^{-k} + n \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathsf{f}',\mathsf{X}',\sigma}(\mathsf{D}')$$

with $\mathsf{D}' \coloneqq \rho(\mathsf{D}, \mathbf{f}, \mathsf{P})$ and $(\mathsf{f}', \mathsf{X}') \coloneqq \psi(\mathbf{f}, \mathsf{P})$.

Proof. The proof is completely analogous to the one of Lemma 4.5.7.

Lemma 4.5.7. *Let n be polynomially bounded. If H is $\mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{1,k}$ RO-CRI secure, then H is also $\mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}$ RO-CRI secure.*

More concretely, let $\mathcal{D}$ denote the set of distinguishers. Then there exists $\rho\colon \mathcal{D} \times \left(\mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}\right) \to \mathcal{D}$ and $\psi\colon \mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k} \to \mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{1,k}$, such that for every $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}$ we have

$$\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) \leq \binom{np}{2} 2^{-k} + n \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathsf{f}',\mathsf{X}',\sigma}(\mathsf{D}')$$

with $\mathsf{D}' \coloneqq \rho(\mathsf{D}, \mathbf{f}, \mathsf{P})$ and $(\mathsf{f}', \mathsf{X}') \coloneqq \psi(\mathbf{f}, \mathsf{P})$.

Proof (Sketch). The proof works along the same lines to the one of Lemma 4.5.8 below; therefore, we only provide a brief sketch. As a first hybrid, we introduce a variant that uses a beacon instead of a random oracle, where a beacon is a resource with the same interface as the random oracle but always answers using fresh randomness even for repeated queries. Distinguishing this hybrid system from the ideal system (that uses the random oracle) can be bounded with the collision probability for the inputs. Since every of the input has k bits of conditional min-entropy, given all previous inputs, the collision for any of them can be bounded with 2^{-k} (cf. the proof below) and there are at most np queries in total. Hence, the total distinction advantage can be bounded by $\binom{np}{2}2^{-k}$.

It remains to bound the distinction advantage between the real system (using the hash function) and our hybrid system (using the

beacon) using the strong-split security for a single message. This can be shown by a simple hybrid-argument with n additional hybrids where the i-th min-entropy seed X_i outputs the i-th message Y_i at interface $\mathtt{A}$ and the messages $Y_1, \ldots, Y_{i-1}$ as additional leakage at interface $\mathtt{E}$. The hybrid then answers the first $i-1$ queries by computing the hash function itself, the i-th message by actually querying the attached system (that uses either the hash function or the beacon), and the remaining queries by uniform random values, simulating the beacon. Defining the resource X' to be the one that chooses uniformly at random among $X_1, \ldots, X_n$ yields the desired bound: $n \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathsf{f}',\mathsf{X}',\boldsymbol{\sigma}_H}(\mathsf{D}')$. □

4.5.5 The Repeated Split-Source Context Set

We now further generalize our strong-split source class, to allow for repeated queries, such as $H\big(hk, H(hk, x||1)||2\big)$. The key idea is to introduce a buffer which stores the results obtained from the hash function. The distinguisher can then choose whether it wants to see those values, or whether it wants to use them as a new query. The filter $\mathbf{f}^{\mathsf{r\text{-}splt}}_{p,r}$ is depicted in Figure 4.11. The parameter r determines the maximal allowed nesting depth. Analogously to the strong-split source, we can then define the $\mathcal{C}^{\mathsf{r\text{-}splt}}_{p,r}$ context set based on this filter as $\mathcal{C}^{\mathsf{r\text{-}splt}}_{p,r} := \{\mathbf{f} \circ \mathbf{f}^{\mathsf{r\text{-}splt}}_{p,r} \mid \mathbf{f} \in \Sigma\} \times \Theta^{\mathsf{ni}}$.

We now prove that strong-split RO-CRI implies repeated-split RO-CRI when furthermore restricted to strict min-entropy seeds. This allows to analyze hash functions only for strong-split security, but use them in contexts where repeated-split security is needed.

Lemma 4.5.8. *Let $k' := \min(k, \log|H.\mathcal{Y}|)$. If H is $\mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k'}$ RO-CRI secure, then H is also $\mathcal{C}^{\mathsf{r\text{-}splt}}_{p,r} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}$ RO-CRI secure.*

More concretely, let $\mathcal{D}$ denote the set of distinguishers. Then there exists a translation of the distinguisher $\rho\colon \mathcal{D} \times \left(\mathcal{C}^{\mathsf{r\text{-}splt}}_{p,r} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}\right) \to \mathcal{D}$ and a translation of the context $\psi\colon \mathcal{C}^{\mathsf{r\text{-}splt}}_{p,r} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k} \to \mathcal{C}^{\mathsf{s\text{-}splt}}_{p} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k'}$, such that for every $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}^{\mathsf{r\text{-}splt}}_{p,r} \cap \mathcal{C}^{\mathsf{s-me}}_{n,k}$ we have

$$\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) \leq \binom{npr}{2} 2^{-(k'-1)} + r \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathsf{f}',\mathsf{X}',\sigma}(\mathsf{D}')$$

with $\mathsf{D}' := \rho(\mathsf{D}, \mathbf{f}, \mathsf{P})$ and $(\mathsf{f}', \mathsf{X}') := \psi(\mathbf{f}, \mathsf{P})$.

Converter $\mathbf{f}^{\text{r-splt}}_{p,r}$

Initialization

$b \leftarrow \bot^p$, $c \leftarrow 0$

Outer Interface

Input: `get`
  $b' \leftarrow b$
  $b \leftarrow \bot^p$
  return b'

Input: $(\texttt{query}, f_1, \ldots, f_p) \in \mathcal{I}^p_{\mathcal{X},\mathcal{A}\to H.\mathcal{X}}$
  $c \leftarrow 1$
  call $v \leftarrow$ `retrieve` at int. A of P
  if v can be parsed as $(x, a) \in \mathcal{X} \times \mathcal{A}$ **then**
    for $i = 1, \ldots, p$ **do**
      $y[i] \leftarrow f_i(x, a)$
    if $\forall i \neq j \colon y[i] \neq y[j]$ **then**
      for $i = 1, \ldots, n$ **do**
        call $b[i] \leftarrow (\texttt{hash}, y[i])$ at int. A of RO_H
    else
      $b \leftarrow \bot^p$
  else
    $b \leftarrow \bot^p$

Input: $(\texttt{repeat}, f_1, \ldots, f_p) \in \mathcal{I}^p_{H.\mathcal{Y}\to H.\mathcal{X}}$
  $c \leftarrow c + 1$
  if $c \leq r \ \wedge \ b \neq \bot^p$ **then**
    for $i = 1, \ldots, p$ **do**
      call $b[i] \leftarrow (\texttt{hash}, f_i(b[i]))$ at int. A of RO_H
  else
    $b \leftarrow \bot^p$

Figure 4.11: The filter $\mathbf{f}^{\text{r-splt}}_{p,r}$ from the repeated split-source context set.

Proof. While this lemma can intuitively be proven using a simple hybrid argument, it turns out to be quite technical. The proof can be found in Appendix A.1. □

4.5.6 The Relation Between ICE and Strong-Split RO-CRI

In this section, we discuss the relation between RO-CRI and the ICE framework introduced in [FM16]. More concretely, we show that statistical-unpredictable ICE security implies strong-split RO-CRI, as

phrased in Theorem 4.5.9. Using this relation between the two frameworks, we especially inherit the random oracle feasibility result from the ICE framework.

The reverse direction, whether strong-split RO-CRI implies some natural notion of ICE, remains an interesting open problem. In general, there seems to be no natural mapping from ICE to RO-CRI. This can be explained by the fundamentally different motivation behind introducing this two generalizations of UCE: ICE tried to allow interaction by making the two stages of UCE more symmetric, whereas RO-CRI exploits the asymmetry of UCE to separate them even further into the protocol of the honest party and the regular distinguisher from indifferentiability.

In terms of random-oracle feasibility, this places RO-CRI as an intermediate notion between the original UCE notion and the stronger ICE notion, while it is still open whether a true separation between those frameworks exists.

Theorem 4.5.9. *Let* H *denote a keyed hash function where the key-space is exponential in the security parameter. If* $\mathsf{H} \in \mathrm{ICE}[\mathcal{C}^{sup}]$*, then* H *is* $\mathcal{C}_p^{\mathsf{s\text{-}splt}} \cap \mathcal{C}_{n,k}^{\mathsf{s-me}}$ *context-restricted indifferentiable from a random oracle for any polynomial* n *and* p*, and* k *such that the guessing probability is negligible.*

Proof. We sketch a proof that for the fixed simulator $\boldsymbol{\sigma}_H$, every context $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}_p^{\mathsf{s\text{-}splt}} \cap \mathcal{C}_{n,k}^{\mathsf{s-me}}$ and distinguisher D can be turned into a pair of equivalent ICE distinguishers D_1 and D_2. Let D_1 internally emulates the distinguisher D and works as follows:

- It initially chooses the hash key hk uniformly at random (as $\boldsymbol{\sigma}_H$) and writes it into the buffer using a WRITE query. This is the only WRITE query D_1 does.
- In every round, it uses obtains the answer from L_2 and passes this to the distinguisher D to obtain the next query. According whether D queries the interface $\mathtt{A}$ with the function f or obtains the leakage at interface $\mathtt{E}$, it produces an appropriate output L_1, either $(\mathtt{A}, f)$ or $(\mathtt{E})$.
- If the distinguisher D outputs the decision bit, D_1 outputs the same bit.

The distinguisher D_2 internally emulates the context $(\mathbf{f}_p^{\mathsf{s\text{-}splt}}, \mathsf{X})$ and works as follows:

- In every round it inspects the value L_1.
  - If L_1 is of the form $(\mathtt{A}, f)$, it passes f to the internal emulation of the context, to obtain the value x that would be queried to the hash function. It then writes x to the buffer and queries the hash function. The resulting value y is returned as L_2.
  - If L_2 is of the form $(\mathtt{E})$, then it queries the interface $\mathtt{E}$ of the internal resource X and returns the result as L_2.
- It always sets $b_2 = 0$.

It is easy to see that the ICE game now behaves exactly the same as the RO-CRI system. Moreover, the queries of D_2 are exactly as unpredictable given the state and randomness of D_1 as are the queries in the RO-CRI system given access to the interface $\mathtt{E}$. Finally, if the hash key hk is unpredictable, then none of the queries of D_1 can be predicted given the complete state and randomness of D_2. This concludes the proof. □

4.6 Split-Security of the Merkle-Damgård Construction

4.6.1 Motivation

Indifferentiability is widely used to prove the security of hash function constructions. Since context-restricted constructions are essentially a refined version of indifferentiability, it is hence natural to consider the RO-CRI security of hash functions as well.

It is easy to show that any hash function construction that is secure according to the traditional indifferentiability notion is also reset-UCE secure if the underlying compression function is reset-UCE secure. On the other hand, for split security no corresponding result has been proven so far. In the following we investigate the split-security of the Merkle-Damgård construction using the RO-CRI framework. While

ideally one could prove that the Merkle-Damgård construction is split secure if the compression function is so, or that the Merkle-Damgård construction is strong-split secure if the compression function is so, we will prove a slightly weaker result:

Consider the Merkle-Damgård construction that splits the message into blocks of length m. We show that the Merkle-Damgård construction is split-secure for inputs having at least one block with k bits of min-entropy, if the compression function is strong-split secure for inputs with $\min(k, m)$ bits of min-entropy.

4.6.2 Formalizing the Theorem

In order for our proof to go through, we require that at least one of the blocks has high min-entropy and not just the overall message has, as in the definition of strict min-entropy seeds. Moreover, we require that this block has k bits of min-entropy given all subsequent blocks. In Lemma 4.6.4 we then show that having a high min-entropy density, i.e., the fraction between the min-entropy and the message length, is a sufficient criteria for this. First, however, let us formally introduce this context set.

Definition 4.6.1. For a block length $\ell \in \mathbb{N}_+$, let Pad_ℓ denote the usual padding scheme of the Merkle-Damgård scheme, that is $\mathrm{Pad}_\ell \colon \{0,1\}^* \to (\{0,1\}^\ell)^+$ that pads a message x by first appending zeros up to a multiple of the block length ℓ, and then appending an encoding of the number of zeros appended as a last block. Moreover, for $X \in \{0,1\}^*$, we denote by X^i the i-th block of $\mathrm{Pad}_\ell(X)$.

Definition 4.6.2. A non-interactive resource is said to be a *k out of ℓ-bit strict min-entropy block*, denoted $\mathsf{P} \in \Theta^{\mathsf{me-blk}}_{k,\ell,b,n}$, if $\mathsf{P} \in \Theta^{\mathsf{seed}}_n$ with $\bigcup_{i \le (b-1)\ell} \{0,1\}^i \times \mathcal{A}$ as the output domain of interface $\mathtt{A}$, and there exist random variables $C_1, \ldots, C_n$ such that $C_i \in \{1, \ldots, \frac{|\mathrm{Pad}_\ell(X_i)|}{\ell}\}$ and

$$\forall i \le n \colon \tilde{H}_\infty\big(X_i^{C_i} \,\big|\, \{X_i^j\}_{j>C_i}, \{X_j\}_{j<i}, \{C_j\}_{j\le i}, \{A_j\}_{j\le i}, Z\big) \;\ge\; k.$$

Moreover, let $\mathcal{C}^{\mathsf{me-blk}}_{k,\ell,b,n} \coloneqq \Sigma \times \Theta^{\mathsf{me-blk}}_{k,\ell,b,n}$.

Remark. Note, that contrary to the classical indifferentiability of the Merkle-Damgård construction, we do not require Pad to be prefix-free:

when combined with the strict min-entropy condition $H(X)$ cannot be extended to $H(\mathrm{Pad}(X)||Y)$, as for $\mathrm{Pad}(X)||Y$ having high min-entropy given X, Y must have so, and thereby the well-known length-extension attack is excluded. Whether a more advanced construction with a finalization function, e.g. HMAC, could be proven secure for a more relaxed context set remains an interesting open problem. We now phrase our main result of this section.

Using the definition of k out of ℓ-bit strict min-entropy block, we can now formally state our theorem about the split-security of the Merkle-Damgård construction.

Theorem 4.6.3. *Let $h\colon \{0,1\}^{m+\ell} \to \{0,1\}^m$ denote a fixed input-length compression function, $H\colon \{0,1\}^* \to \{0,1\}^m$ denote the hash function obtained by first padding the message using Pad_ℓ and then applying the Merkle-Damgård scheme using h, and let $k' \coloneqq \min(k,m)$. Then, if h is $\mathcal{C}_1^{\mathsf{s\text{-}splt}} \cap \mathcal{C}_{1,k'}^{\mathsf{s-me}}$ RO-CRI secure, then H is $\mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}_{k,\ell,b,n}^{\mathsf{me-blk}}$ RO-CRI secure for any polynomial b and n.*

More explicitly, there exists $\rho_1, \rho_2\colon \mathcal{D} \times \left(\mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}_{k,\ell,b,n}^{\mathsf{me-blk}}\right) \to \mathcal{D}$ and $\psi_1, \psi_2\colon \mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}_{k,\ell,b,n}^{\mathsf{me-blk}} \to \mathcal{C}_1^{\mathsf{s\text{-}splt}} \cap \mathcal{C}_{1,k'}^{\mathsf{s-me}}$ such that for all distinguishers D and all contexts $(\mathbf{f},\mathsf{P}) \in \mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}_{k,\ell,b,n}^{\mathsf{me-blk}}$ we have

$$\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) \leq \binom{n}{2} \cdot 2^{-k} + n \cdot \binom{b}{2} \cdot 2^{-(k'-1)} + nb \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{h},\mathsf{f}',\mathsf{X}',\sigma'}(\mathsf{D}') + n \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{h},\mathsf{f}'',\mathsf{X}'',\sigma''}(\mathsf{D}'')$$

with $\mathsf{D}' \coloneqq \rho_1(\mathsf{D},\mathbf{f},\mathsf{P})$, $\mathsf{D}'' \coloneqq \rho_2(\mathsf{D},\mathbf{f},\mathsf{P})$, $(\mathsf{f}',\mathsf{X}') \coloneqq \psi_1(\mathbf{f},\mathsf{P})$, $(\mathsf{f}'',\mathsf{X}'') \coloneqq \psi_2(\mathbf{f},\mathsf{P})$, and σ' and σ'' denoting slightly modified variants of σ.

4.6.3 Proof of Theorem 4.6.3

Let us first provide an intuitive argument for the case of a single message. Assume that the message y being hash by the Merkle-Damgård scheme is split into b blocks, out of which at least one has k bits of min-entropy. Let c denote the index of this block, i.e., y_c has at least k bits of min-entropy. Hence, according to our assumption on the compression function, the output q of this block cannot be distinguished from the output of a random oracle, as depicted in Figure 4.12. Given that this

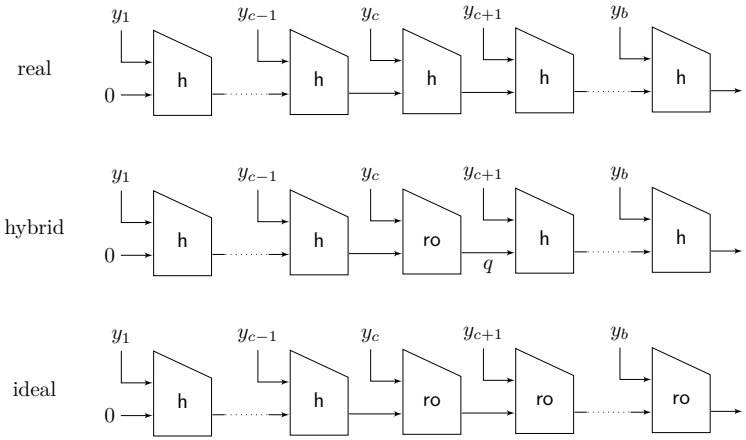


Figure 4.12: The real and the ideal setting for the Merkle-Damgård construction if block c has high min-entropy.

output is just a uniformly random value of length m, by induction, neither can be the output of any subsequent block be distinguished from the output of a random oracle. Therefore, the final output cannot be distinguished from the uniform random value $RO(X)$. We now proceed with the formal proof of Theorem 4.6.3.

Proof of Theorem 4.6.3. Using Lemma 4.5.6 it suffices to show

$$\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) \le \binom{b}{2} \cdot 2^{-(k'-1)} + b \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{h},\mathsf{f}',\mathsf{X}',\sigma'}(\mathsf{D}') + \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{h},\mathsf{f}'',\mathsf{X}'',\sigma''}(\mathsf{D}'')$$

for all distinguishers D and all contexts $(\mathbf{f},\mathsf{P}) \in \mathcal{C}^{\mathsf{splt}} \cap \mathcal{C}^{\mathsf{me-blk}}_{k,\ell,1}$.

Next, observe that for any message $y \in \bigcup_{i \le (b-1)\ell} \{0,1\}^i$, applying the padding Pad_ℓ results in a message that has at most b blocks. Without loss of generality, we assume in the following that there are always exactly b blocks.

Given any k out of ℓ-bit min-entropy block seed P with a single output, we first introduce two k'-bit min-entropy seeds X' and X''. Note that the function ψ_1 and ψ_2 are just mappings from one context to

another one relating the two problems and, in contrast to the reduction translating the distinguisher, do not need to be efficiently computable. Therefore, it is sufficient to know that such a random variable C from Definition 4.6.2 exists for the seed X.

Definition of X':

Let X' denote the resource that samples (y, z) using the same distribution as P, applies the padding, and splits it into the blocks $y_1, \ldots, y_b$. Then, it sample the random variable C to obtain the index c. Finally, it outputs the pair (a', y') with $a' = (y_0, y_1, \ldots, y_{c-1})$ and $y' = y_c$ at interface $\mathtt{A}$ and $z' = (z, c, y_{c+1}, \ldots, y_b)$ at interface $\mathtt{E}$.

Definition of X'':

Let X'' denote the resource that samples (y, z) using the same distribution as P, applies the padding, and splits it into the blocks $y_1, \ldots, y_b$. Then, it sample the random variable C to obtain the index c and chooses $q \in \{0,1\}^m$ uniformly at random, outputs the pair $(a', y') \coloneqq (\bot, q)$ at interface $\mathtt{A}$, and the value $z' \coloneqq (z, c, y_{c+1}, \ldots, y_b)$ at interface $\mathtt{E}$.

Observe that X' is a $k \geq k'$ bit (strict) min-entropy seed, since X is k out of n-bit min-entropy block seed. Similarly, since q is chosen independently of all other random variables, the seed X'' is a $m \geq k'$ bit strict min-entropy seed. Moreover, both of them output only a single value, i.e., $\mathsf{X}', \mathsf{X}'' \in \Theta^{\mathsf{s-me}}_{1,k'}$.

Next, we briefly sketch the two simulators σ' and σ'': they both internally run σ. Whenever σ request for the leakage z of the seed, they query the leakage z' at the corresponding inner interface and return the first component z to σ.

Now, we introduce two converter systems C' and C'' that at the inside interface connect to both the interface $\mathtt{A}$ and the interface $\mathtt{E}$ of the connected system, and at the outside interface emulates both the interfaces as well.

The system C' works as follows:

First it obtains hk and $z' = (z, c, y_{c+1}, \ldots, y_b)$ at the interfaces $\mathtt{E.H}$ and $\mathtt{E.X}$ of the connected system. When receiving the input $\mathtt{retrieve}$ at the outside interface $\mathtt{A}$, it outputs $(\mathtt{retrieve}, f)$

at the inside interface A, where f is the function that on input (y_c, a') first splits $a' = (y_0, \ldots, y_{c-1})$, then computes the prefix $p = h_{hk}(\ldots h_{hk}(h_{hk}(0||y_0)||y_1)\ldots||y_{c-1})$, and finally returns $p||y_c$. Since both p and y_c are of fixed length, this function is injective in the first argument. When obtaining the returned value y', it then computes the suffix $s = h_{hk}(\ldots h_{hk}(h_{hk}(y'||y_{c+1})||y_{c+2})\ldots||y_b)$ and returns s at the outside interface A. When receiving the input retrieve at either the interface E.H or E.X it returns hk or z, respectively.

The system C'' works as follows:

First it obtains hk and $z' = (z, c, x_{c+1}, \ldots, x_b)$ at the interfaces E.H and E.X of the connected system. When receiving the input retrieve at the outside interface A, it first outputs $(\texttt{query}, f)$ at the inside interface A, where f is the function that on input $(q, \bot)$ returns $q||y_{c+1}$. This function is injective in the first argument. Then, for $i = c+2, \ldots, b$ it outputs $(\texttt{repeat}, f)$ at the inside interface A, where f is the function that on input (x) returns $x||y_i$. Finally, it outputs get at the inside interface A and returns the obtained value at the outside interface A. When receiving the input retrieve at either the interface E.H or E.R it returns hk or z, respectively.

It is easy to verify, that the composed system $\mathsf{C}'\mathbf{f}_p^{\mathsf{s\text{-}splt}}[\mathsf{h}, \mathsf{X}']$ at the interface A outputs $H(y)$ and, thus, we have the equivalence $\mathbf{f}^{\mathsf{splt}}[\mathsf{H}, \mathsf{P}] \equiv \mathsf{C}'\mathbf{f}_p^{\mathsf{s\text{-}splt}}[\mathsf{h}, \mathsf{X}']$. In the following $\mathbf{f}_{1,p}^{\mathsf{r\text{-}splt}}$ denote the filter introduced in Section 4.5.5. It is then easy to verify that the final output of the composed system $\mathsf{C}''\mathbf{f}_{1,p}^{\mathsf{r\text{-}splt}}[\mathsf{ro}, \mathsf{X}'']\boldsymbol{\sigma}''$ at the interface A is just a uniform random value independent of hk and z. Hence, this system behaves equivalently to $\mathbf{f}^{\mathsf{splt}}[\mathsf{RO}, \mathsf{X}]\boldsymbol{\sigma}$ that outputs a single uniform random value as well. In short, we have $\mathbf{f}^{\mathsf{splt}}[\mathsf{RO}, \mathsf{X}]\boldsymbol{\sigma} \equiv \mathsf{C}''\mathbf{f}_{1,p}^{\mathsf{r\text{-}splt}}[\mathsf{ro}, \mathsf{X}'']\boldsymbol{\sigma}''$.

Using those two equivalences, and by introducing two hybrids $\mathsf{C}'\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{ro}, \mathsf{X}']\boldsymbol{\sigma}'$ and $\mathsf{C}''\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{h}, \mathsf{X}''\boldsymbol{\sigma}''$, we can rewrite the distinction advantage as:

$$\begin{aligned}\Delta^{\mathsf{D}}(\mathbf{f}^{\mathsf{splt}}[\mathsf{H}, \mathsf{P}], \mathbf{f}^{\mathsf{splt}}[\mathsf{RO}, \mathsf{P}]\boldsymbol{\sigma}) = {} & \Delta^{\mathsf{D}}(\mathsf{C}'\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{h}, \mathsf{X}'], \mathsf{C}'\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{ro}, \mathsf{X}']\boldsymbol{\sigma}') \\ & + \Delta^{\mathsf{D}}(\mathsf{C}'\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{ro}, \mathsf{X}']\sigma', \mathsf{C}''\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{h}, \mathsf{X}'']) \\ & + \Delta^{\mathsf{D}}(\mathsf{C}''\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{h}, \mathsf{X}''], \mathsf{C}''\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{ro}, \mathsf{X}'']\boldsymbol{\sigma}'').\end{aligned}$$

Next, observe that the systems $\mathsf{C}'\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{ro},\mathsf{X}']\boldsymbol{\sigma}'$ and $\mathsf{C}''\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{h},\mathsf{X}'']\boldsymbol{\sigma}''$ both implement exactly the same hybrid system depicted in Figure 4.12: The system $\mathsf{C}'\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{ro},\mathsf{X}']\boldsymbol{\sigma}'$ actually computes this value by first using the compression function h on the blocks 1 to $c-1$, then uses the fixed input size random oracle on the block c, and finishes by using h on the remaining blocks. However, note that the value output by ro is just a uniform random value, as ro is private and not used beside this one query. The system $\mathsf{C}''\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{h},\mathsf{X}'']$ skips the initial computes and chooses q uniformly at random (in X''). As a result, we can simplify the distinction advantage to

$$\Delta^{\mathsf{D}}(\mathbf{f}^{\mathsf{splt}}[\mathsf{H},\mathsf{P}],\mathbf{f}^{\mathsf{splt}}[\mathsf{RO},\mathsf{P}]\boldsymbol{\sigma}) = \Delta^{\mathsf{DC}'}(\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{h},\mathsf{X}'],\mathbf{f}_1^{\mathsf{s\text{-}splt}}[\mathsf{ro},\mathsf{X}']\boldsymbol{\sigma}') + \Delta^{\mathsf{DC}''}(\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{h},\mathsf{X}''],\mathbf{f}_{1,b}^{\mathsf{r\text{-}splt}}[\mathsf{ro},\mathsf{X}'']\boldsymbol{\sigma}'').$$

Applying the definition of $\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D})$, as well as Lemmas 4.5.3 and 4.5.8 (the latter with $r = b$, $n = 1$, and $p = 1$), concludes the proof. □

4.6.4 A Sufficient Condition Based on Min-Entropy Splitting

To conclude this section, we now present a sufficient condition for a seed to satisfy Definition 4.6.2 based on the length of the message and its overall min-entropy. More concretely, we prove that if a message is split into b blocks of size n, and has overall min-entropy of k bits, then there exists a block with $\frac{k}{b} - \log_2(b)$ bits of min-entropy, given all succeeding blocks. In order to more closely resembles the chain rule of Shannon entropy, the proposition is stated with conditioning on all preceding message $X_1 \dots X_{C-1}$ instead of all succeeding ones. The converse result can easily be obtained by simply relabeling the blocks.

Lemma 4.6.4. *Let $X_1, \dots, X_b$ and Z be random variables (over possibly different alphabets) with $\tilde{H}_\infty(X_1 \dots X_b \mid Z) \geq k$. Then, there exists a random variable C over the set $\{1, \dots, b\}$ such that*

$$\tilde{H}_\infty(X_C \mid X_1 \dots X_{C-1} C Z) \geq \tfrac{k}{b} - \log_2(b).$$

Proof. Let $Y_C \coloneqq (X_1, \dots, X_{C-1})$, with Y_0 denoting the empty string λ.

Second, let for every z in the support of Z,

$$p_z \coloneqq \max_{x_1,\ldots,x_b} \mathsf{P}_{X_1\ldots X_b|Z}(x_1,\ldots,x_b,z),$$

that is, $\tilde{H}_\infty(X_1\ldots X_b \,|\, Z) = -\log \mathbb{E}_z[p_z]$. Moreover, once C is defined (see below), let

$$q_z \coloneqq \mathbb{E}_{c,y}[\max_x \mathsf{P}_{X_C|CY_CZ}(x,c,y,z) \,|\, Z = z]$$

and note that $\tilde{H}_\infty(X_C \,|\, CY_CZ) = -\log \mathbb{E}_z[q_z]$. We now proceed by showing that for all z, $q_z \leq b \cdot p_z^{1/b}$. To this end, we extend the probability distribution $\mathsf{P}_{X_1\ldots X_bZ}$ by defining the random variable C as follows:

$$C = \begin{cases} 1 & \text{if } \mathsf{P}_{X_1|Z}(x_1,z) < p_z^{1/b} \\ 2 & \text{else if } \mathsf{P}_{X_1X_2|Z}(x_1,x_2,z) < p_z^{2/b} \\ \vdots & \\ b-1 & \text{else if } \mathsf{P}_{X_1\ldots X_{k-1}|Z}(x_1,\ldots,x_{b-1},z) < p_z^{(b-1)/k} \\ b & \text{else.} \end{cases}$$

Observe that with

$$\begin{aligned} \mathcal{Y}_{c,z} &\coloneqq \{y \mid \mathsf{P}_{CY_C|Z}(c,y,z) > 0\} \\ \mathcal{X}_{c,z,y} &\coloneqq \{x \mid \mathsf{P}_{X_CCY_C|Z}(x,c,y,z) > 0\} \end{aligned}$$

we can bound q_z as follows:

$$\begin{aligned}
q_z &= \mathbb{E}_{c,y}[\max_x \mathsf{P}_{X_C|CY_CZ}(x,c,y,z) \mid Z=z] \\
&= \sum_{c=1}^{b} \sum_{y\in\mathcal{Y}_{c,z}} \mathsf{P}_{CY_C|Z}(c,y,z) \cdot \max_{x\in\mathcal{X}_{c,z,y}} \mathsf{P}_{X_C|CY_CZ}(x,c,y,z) \\
&= \sum_{c=1}^{b} \sum_{y\in\mathcal{Y}_{c,z}} \mathsf{P}_{CY_C|Z}(c,y,z) \cdot \max_{x\in\mathcal{X}_{c,z,y}} \frac{\mathsf{P}_{X_CCY_C|Z}(x,c,y,z)}{\mathsf{P}_{CY_C|Z}(c,y,z)} \\
&= \sum_{c=1}^{b} \sum_{y\in\mathcal{Y}_{c,z}} \max_{x\in\mathcal{X}_{c,z,y}} \mathsf{P}_{X_CCY_C|Z}(x,c,y,z) \\
&\leq \sum_{c=1}^{b} \sum_{y\in\mathcal{Y}_{c,z}} \max_{x\in\mathcal{X}_{c,z,y}} \mathsf{P}_{X_CY_C|Z}(x,y,z).
\end{aligned}$$

We now further bound this term using a case distinction on c. First, consider the case $c=1$. Since $Y_1=\lambda$ is constant, we have $\mathcal{Y}_{1,z}\subseteq\{\lambda\}$ and $\mathsf{P}_{X_1Y_1|Z}(x,\lambda,z)=\mathsf{P}_{X_1|Z}(x,z)$. Moreover, $x\in\mathcal{X}_{1,z,\lambda}$ implies $\mathsf{P}_{X_1C|Z}(x,1,z)>0$, which by the definition of C in turn implies $\mathsf{P}_{X_1|Z}(x,z)<p_z^{1/b}$. Hence

$$\sum_{y\in\mathcal{Y}_{1,z}} \max_{x\in\mathcal{X}_{1,z,y}} \mathsf{P}_{X_1Y_1|Z}(x,y,z) \leq \max_{x\in\mathcal{X}_{1,z,\lambda}} \mathsf{P}_{X_1|Z}(x,z) \leq p_z^{1/b}.$$

For all $i\in\{2,\ldots,b-1\}$ observe that by the definition of C we have that $\mathcal{X}_{i,z,y}\subseteq\left\{x \mid \mathsf{P}_{X_iY_i|Z}(x,y,z)<p_z^{i/b}\right\}$ and $\mathcal{Y}_{i,z}\subseteq\left\{y \mid \mathsf{P}_{Y_i|Z}(y,z)\geq p_z^{(i-1)/b}\right\}$. From the latter we can conclude that $|\mathcal{Y}_{i,z}|\leq\frac{1}{p_z^{(i-1)/b}}$ and, hence, we obtain

$$\sum_{y\in\mathcal{Y}_{i,z}} \max_{x\in\mathcal{X}_{i,z,y}} \mathsf{P}_{X_iY_i|Z}(x,y,z) \leq \sum_{y\in\mathcal{Y}_{i,z}} p_z^{i/b} \leq \frac{p_z^{i/b}}{p_z^{(i-1)/b}} = p_z^{1/b}.$$

Finally, for $c=b$, we have that $\mathcal{Y}_{b,z}\subseteq\{y \mid \mathsf{P}_{Y_b|Z}(y,z)\geq p_z^{(b-1)/b}\}$ and, therefore, we obtain $|\mathcal{Y}_{b,z}|\leq\frac{1}{p_z^{(b-1)/b}}$. Using the definition of $Y_b=(X_1,\ldots,X_{b-1})$ and p_z, we get $\max_{x\in\mathcal{X}_{b,z,y}}\mathsf{P}_{X_bY_b|Z}(x,y,z)\leq p_z$

for every $y = (x_1, \ldots, x_{b-1})$. Hence,

$$\sum_{y \in \mathcal{Y}_{b,z}} \max_{x \in \mathcal{X}_{b,z,y}} \mathsf{P}_{X_b Y_b | Z}(x, y, z) \leq \sum_{y \in \mathcal{Y}_{b,z}} p_z \leq \frac{p_z}{p_z^{(b-1)/b}} = p_z^{1/b}$$

as well. In summary,

$$\begin{aligned} q_z &= \mathbb{E}_{c,y}[\max_x \mathsf{P}_{X_C | C Y_C Z}(x, c, y, z) \mid Z = z] \\ &\leq \sum_{c=1}^{b} \sum_{y \in \mathcal{Y}_{c,z}} \max_{x \in \mathcal{X}_{c,z,y}} \mathsf{P}_{X_C Y_C | Z}(x, y, z) \\ &\leq \sum_{c=1}^{b} p_z^{1/b} \\ &\leq b \cdot p_z^{1/b} \end{aligned}$$

Using the monotonicity of the expected value, Jensen's inequality, and the assumed inequality $\tilde{H}_\infty(X_1 \ldots X_b \mid Z) \geq k$ yields

$$\begin{aligned} 2^{-\tilde{H}_\infty(X_C | C Y_C Z)} &= \mathbb{E}_z[q_z] \leq \mathbb{E}_z[b \cdot p_z^{1/b}] \leq b \cdot \mathbb{E}_z[p_z]^{1/b} \\ &= b \cdot \left(2^{-\tilde{H}_\infty(X_1 \ldots X_b | Z)}\right)^{1/b} \leq 2^{\log b} \cdot 2^{-k/b} = 2^{-(k/b - \log b)} \end{aligned}$$

concluding the proof. □

This lemma is a generalization of the randomized chain rule proven by the authors of [DFR+07] (similar variants exists also in [BK12; Wul07]) stating that there exists a binary random variable C such that $H_\infty(X_{1-C} C) \geq H_\infty(X_0 X_1)/2$. Note that the main difference of our result is, that it conditions on all previous blocks, i.e., it essentially represents the min-entropy equivalence of the strong chain rule $H(X_0) + H(X_1 \mid X_0) = H(X_0 X_1)$ instead of $H(X_0) + H(X_1) \geq H(X_0 X_1)$.

Chapter 5

Overcoming the Commitment Problem

In this chapter, we propose a novel type of specifications that overcomes the so-called simulator commitment problem. More concretely, we introduce specifications that formalize interval-wise guarantees, i.e., guarantees that hold in between two events. The specifications are formalized as relaxations of an (overly) idealized one, where the relaxation admits arbitrary behavior, i.e., waives all guarantees, outside the interval.

5.1 Introduction

5.1.1 Motivation

One of the most fundamental obstacle hindering the adoption of composable security definitions is the so-called simulator commitment problem, or commitment problem for short. It mainly arises when considering adaptive security. In a nutshell, it describes the simulator's inability to explain some of its previous choices the moment a party gets corrupted. More concretely, consider the example of two parties securing their communication using symmetric encryption. The intuition is that the adversary does not learn the messages until either of the parties gets

corrupted, thereby revealing the key. Before, the adversary should learn at most the length. As a result, the simulator, in the first phase, has to output a fake ciphertexts independent of the real messages. For any semantically secure encryption scheme he can actually do so. This, however, commits him on those fake ciphertexts. At the moment a party gets corrupted, the simulator then needs to be able to explain those ciphertexts by outputting a matching encryption key. Even if he learns all the previous messages, he will not be able to do so for regular encryption schemes. Note, however, that the commitment problem is not restricted to adaptive corruptions only. Similar issues also arise, for instance, in the context of password-based security [DGMT17] or identity-based encryption [HMM15], where it has been shown that due to this commitment problem the standard game-based notions do not induce the expected corresponding composable statements.

On a general level, this raises the fundamental question whether such impossibility results actually indicate a security issue, and hence protocols not satisfying the stronger composable definitions should not be used, or whether they present an artifact of the framework. Especially for the commitment problem, the common understanding is that the latter is true. As a consequence, the commitment problem is commonly dealt with by either reverting to composable security with static corruptions only, or by simply retracting to standalone game-based definitions. Alternatively, in work where modularity and composability is crucial, one often simply accepts (unrealistic) strong setup assumptions, such as a common reference string (CRS) to avoid the commitment problem. Moreover, the corresponding protocols are often less efficient than their standalone counterparts.

As a result, a number of approaches to overcome the issue have been proposed [CK02; Pas03; PS04; BDHK06; BDH+17]. The majority of them, however, introduces an intermediate notion (between standalone security and the traditional simulation-based composable security notion). Thus, to achieve those notions, one still relies on strong setup assumptions and/or less efficient protocols, when compared to the classical standalone secure protocols. Moreover, most existing approaches are intrinsically motivated from a technical point of view—weakening the simulation-based notion (e.g., UC) sufficiently until the impossibilities can be evaded—leading to rather technical definitions with sometimes unclear semantics.

Ultimately, this poses the question: how can we express the security properties achieved by protocols for which so far only standalone security notions are known composably?

5.1.2 Contributions

Interval-wise guarantees. In this work, we propose an alternative solution to the simulator-commitment problem that is aimed at expressing the guarantees of regular schemes within a composable framework. More concretely, we introduce a novel type of specification that avoids the commitment problem while providing a number of distinct benefits. First, it provides a clean semantics of how the guarantees should be interpreted. Second, it holds in any environment, just as any statement in the CC framework. Third, it is equipped with a composition theorem.

Since the commitment problem usually occurs at a very specific point of the protocol execution, such as when a party gets corrupted, where the security guarantees of the protocol anyway inherently change, our novel specification notion is centered around the very natural idea of formalizing guarantees that hold in a certain interval (between two events). That is, our notion for instance allows to formalize separate security guarantees before and after the corruption event. In contrast to existing simulation-based notions, we thereby only require the simulation to work within each interval, not forcing the simulation to be consistent between the intervals (which causes the initial commitment issues). We discuss how the security guarantees provided by our notion should be interpreted, when stronger notions might still be desirable, and how our notion fits into the space of static versus adaptive security.

On a technical level, we formalize interval-wise guarantees as a novel type of relaxation of an (overly) idealized specification. We carefully consider the subtleties arising when defining these relaxations and show how they interact with the other aspects of the framework. Finally, we present the respective composition theorem, that actually supersedes all the existing ones, and in particular allows to syntactically combine multiple such interval-wise construction statements, or an interval-wise one with a regular construction statement.

Applications. As a third contribution, we apply our methodology to several examples. First, we consider the encrypt-then-MAC paradigm in

a setting where the keys can adaptively leak to the adversary, stylizing adaptive passive corruptions. Using our interval-wise guarantees, we obtain a simple composable security definition thereof without the need for non-committing encryption. More concretely, we consider the following three properties. First, we require the messages to be confidential as long as neither the encryption nor the authentication key leaked. (An IND-CPA secure scheme cannot guarantee confidentiality without authenticity.) In our definition, this is phrased as the construction of a secure channel *up to that point*. Second, between the exposure of the encryption key and the authentication key, we require communication to still be authentic, i.e., an authenticated channel to be constructed. Finally, after the encryption key has been exposed, we still require correctness.

As a second application, we present a composable formalization of information-theoretically binding commitment schemes realizable in the plain model. We then show how, based on such a commitment scheme, Blum's protocol constructs a composable coin-toss notion. Applying composition then directly implies that this formalization can be achieved in the plain model as well. While the resulting specification is obviously too weak to serve as a common reference string, it guarantees unbiasedness. Hence, it is provides a good enough type of randomness resource whenever unbiasedness is sufficient, in particular formalizing and formally validating the intuitive-level argumentation about flipping a coin over the telephone of the corresponding papers of that time.

Finally, we consider the composable guarantees of identity-based encryption. We revisit the result by Hofheinz, Matt, and Maurer [HMM15] that shows the standard ind-id-cpa notion to be too weak when considering a traditional composable statement based on the existence of a single simulator, even when considering *static corruptions*, due to the commitment problem. Furthermore, the authors have shown that the same weaker construction that actually can be achieved, could also be achieved by a weaker game-based notion ind-id1-cpa, modeling so-called lunch-time attacks. We refute their results in the following way: Based on interval-wise guarantees we formalize a composable specification of IBE that corresponds exactly to the standard ind-id-cpa notion.

5.1.3 Related Work

A number of approaches have been proposed in order to circumvent the aforementioned issues of composable security. First, Canetti and Krawczyk proposed the notion of non-information oracles [CK02] within the UC-framework. A non-information oracle is essentially a game-based definition embedded into an ideal functionality. For instance, rather than saying that an encryption scheme should realize a secure channel that only leaks the length, the respective functionality leaks the output of the non-information oracle, which is required to satisfy a CPA-like definition. While this circumvents the commitment problem, there are two drawbacks. First, it weakens composition by requiring explicit reductions to the embedded games in the security proof of the higher-level protocols using the functionality. Second, for each ideal functionality a different type of non-information oracle needs to be defined. As a consequence, the question of the "right" non-information oracle re-arises, just like when defining a security game.

Second, a line of work considers super-polynomial simulators [Pas03; PS04; BDH+17]. The initial proposal by Pass [Pas03] considered sub-exponential simulators and polynomially bounded environments. This implies, however, that the simulator cannot be absorbed into the environment, ceding some of the most fundamental composition properties of the UC-framework. The later works by Prabhakaran and Sahai [PS04] and Broadnax et al. [BDH+17] empower the simulator in a more controlled manner, preserving most natural composition properties. Their adoption, however, still suffers from being rather technical, and moreover, still quite limited in the number of issues a more powerful simulator can overcome. For instance, when considering a PRG whose seed might leak, even an all powerful simulator will not be able to explain a truly randomly chosen output with an appropriate seed.

Finally, Backes, Dürmuth, Hofheinz, and Küsters [BDHK06] proposed an approach where the real-world resource would just disallow certain activation sequences by the environment that were otherwise impossible to simulate. While this avoids the complications of the other approaches, it scarifies the evident semantics of composable security notions by excluding certain—deemed artificial—attacks. A similar approach has recently been used by Jost, Maurer and Mularczyk in [JMM19b].

5.1.4 The Constructive Cryptography Setting

This chapter is based on Constructive Cryptography with events, as introduce in Chapter 3. Other more concrete aspects, such as the set of involved parties, are specified for each specific example separately.

5.2 Interval-Wise Guarantees: Motivation and Intuition

In this section, we outline the general approach, and its motivation, proposed in this work, before we deep dive into the technicalities in Section 5.3. In particular, we believe that the conceptual contributions are of interest independent from the exact mathematical formalization.

5.2.1 A Motivating Example

Recall the example from Section 2.2.6, where two parties Alice and Bob, wanted to communicate authentically over the Internet. For this section, assume that they also want to achieve confidentiality. If they have a pre-shared secret key available, e.g. from running a key agreement protocol, then it is well known that the encrypt-then-MAC paradigm achieves the desired goal. Assuming independent keys for the encryption and MAC scheme, this construction is secure if the underlying encryption scheme is IND-CPA secure and the MAC scheme is weakly unforgeable.

What, however, if we assume that in reality the keys to not be one hundred percent secure? Intuitively one should expect the scheme to remain secure until either of the keys leak to an adversary, and the security properties then to gracefully downgrade accordingly. More concretely, there is little reason to doubt the following security guarantees should be provided by the scheme:

1. until either of the keys leak, the scheme should provide both confidentiality and authenticity;

2. if only the encryption key leaked so far, then the scheme should still provide authenticity;

3. once the MAC key leaked, the scheme should at least still provide correctness, i.e., allow the parties to communicate in the absence of an active network attack.

(Note that if first the MAC key gets exposed, then a scheme that is only IND-CPA secure might not provide full confidentiality.)

5.2.2 A Naive Attempt

While the encrypt-then-MAC paradigm has composably proven to be sound in a context where both parties are honest and the keys are secure (e.g. [CK02; MT10]), extending those results to deal with key exposures has turned out to be surprisingly strenuous.

Intuitively, one might model the achieved security guarantees as an *secure channel with downgradable security*, which waives confidentiality and authenticity once the respective keys leaked. The protocol should then construct such a channel from an insecure channel and two leakable keys, for authentication and encryption, respectively. To this end, we extend the resources introduced in Section 2.2.6 accordingly. First, we now assume two leakable keys $\mathsf{AuthKey}$ and EncKey, rather than the perfectly secure one in Section 2.2.6. Second, we now consider an authentic channel $\mathsf{AuthChDg}$ with downgradable security in response to the key-leakage events. Analogously, we consider a secure channel $\mathsf{SecChDg}$ with downgradable security, and finally we assume an insecure channel $\mathsf{InsecCh}$. A formal description of the resources can be found in Figure 5.1.

First, consider authentication, i.e., the construction of $\mathsf{AuthChDg}$ from $\mathsf{InsecCh}$ and $\mathsf{AuthKey}$. Indeed, one can show the following result.

Proposition 5.2.1. *Let* $\mathsf{AuthChDg}$ *denote the authenticated channel that degrades its security once the respective key is leaked, as formally defined in Figure 5.1. Then, there exists a simulator* $\boldsymbol{\sigma}_{\mathsf{MAC}}$ *such that*

$$\left[\mathsf{AuthKey}, \mathsf{InsecCh}\right] \xrightarrow[\mathsf{sim}]{\boldsymbol{\pi}_{\mathsf{MAC}}, \boldsymbol{\sigma}_{\mathsf{MAC}}, \epsilon_{\mathsf{MAC}}} \mathsf{AuthChDg},$$

where ϵ_{MAC} *denotes a simple reduction to the MAC-forgery game.*

Proof. This is a well-known result, which has for instance been sketched in [Mau11].

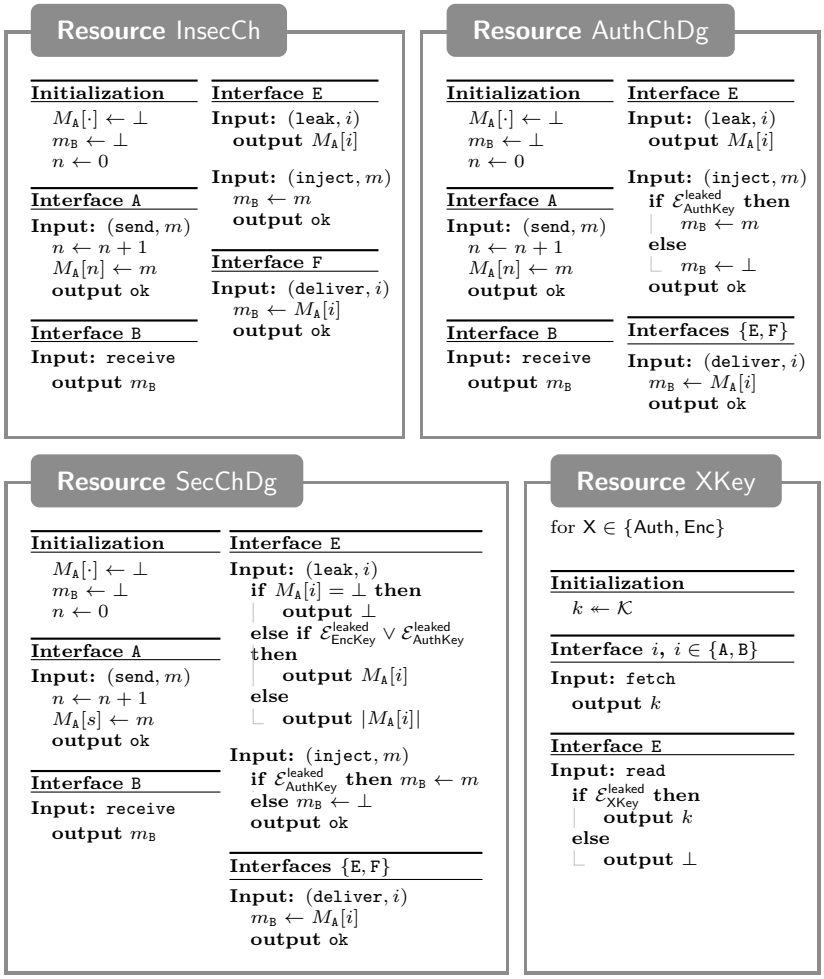


Figure 5.1: The resources involved in the encrypt-then-MAC example. Observe how the authenticated and the secure channel degrade their guarantees once the respective keys have been leaked.

However, once we turn our attention towards the construction step from Section 2.2.6—using encryption to achieve confidentiality—we run into the so-called simulator commitment problem of composable security when considering leakable keys. This is expressed by the following proposition.

Proposition 5.2.2. *Let* $\mathsf{SecChDg}$ *denote a secure channel that degrades the respective guarantees once the keys have been exposed, as depicted in Figure 5.1. For any (efficient) simulator* $\boldsymbol{\sigma}_{\mathsf{ENC}}$*, and reduction* ϵ_{CPA} *to the IND-CPA game, we have*

$$\big[\mathsf{EncKey}, \mathsf{AuthChDg}\big] \xrightarrow[\mathsf{sim}]{\boldsymbol{\pi}_{\mathsf{ENC}}, \boldsymbol{\sigma}_{\mathsf{ENC}}, \epsilon_{\mathsf{CPA}}} \mathsf{SecChDg},$$

i.e., IND-CPA security does not suffice.

Proof (Sketch). In the first phase, the simulator has to produce, without knowing the message, fake ciphertexts $c_1, c_2, \ldots, c_n$ that look indistinguishable from the real one. For an IND-CPA secure scheme, he can easily do so by encrypting an arbitrary message of the correct length. The moment the encryption key leaks in the real world, he however has to output a uniformly looking key that makes his ciphertexts decrypt to the correct messages. Even knowing the messages by now, this is infeasible unless we assume a non-committing encryption scheme. Furthermore, as long as the requested key is shorter than n, Nielsen [Nie02a] showed that NCE cannot be achieved in the standard model by non-interactive protocols. □

Of course one could avoid this impossibility by utilizing stronger primitives and/or assumptions, such as non-committing encryption. In some contexts, such as when considering deniability, their stronger guarantees might even be inherently necessary. In this work, we however pose the following question: How can we express the aforementioned security guarantees, that the encrypt-then-MAC paradigm using regular encryption intuitively does provide, in a composable framework? That is, rather than establishing a stronger security notion, we aim at expressing the exact guarantees provided by existing game-based notions.

5.2.3 Our Solution

So how to express the natural properties that are achieved? First, let us have another look at the reason for the impossibility: traditional simulation-based security notions require the simulator to commit to a ciphertext, emulating the encryption, based on the length only. Even if the simulator later gets to learn the entire message, it cannot come up with an encryption key that decrypts the previously output ciphertext to this message. Observe, however, that outputting the length only is just a technical way of expressing confidentiality until either one of the keys leak. In principle, there is no inherent requirement for a consistent simulation strategy across the different phases of the experiments. This is exactly what our proposal of interval-wise guarantees builds on: allowing disjoint simulation strategies for different phases of a protocol run. In other words, we simply make three disjoint security statements, one guaranteeing confidentiality and authenticity until either key is leaked, one only guaranteeing authenticity between the exposure of the encryption key and the MAC key, and one guaranteeing correct delivery of messages afterwards. Given the specification centric approach of Constructive Cryptography, this can be phrased as

$$\boldsymbol{\pi}_{\mathsf{ENC}}\boldsymbol{\pi}_{\mathsf{MAC}}[\mathsf{AuthKey}, \mathsf{EncKey}, \mathsf{InsecCh}] \subseteq \mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3,$$

where $\mathcal{S}_1$ to $\mathcal{S}_3$ are specifications formalizing the respective guarantees.

Phrasing separate statements can trivially be done in any framework, but also comes with a number of drawbacks. First, having to specify three constructions of unconnected, potentially differently described, specifications incurs a certain cognitive overhead, making the overall achieved security more demanding to understand. Second, and more severely, one loses some compositional properties. In particular, the analysis of another protocol building on top of those guarantees would require to make the exact same case distinction.

To overcome those drawbacks, we phrase each guarantee as an appropriate *interval-wise relaxation* of the same underlying resource: the downgradable secure channel. That is, we phrase security as

$$\begin{aligned}\boldsymbol{\pi}_{\mathsf{ENC}}\boldsymbol{\pi}_{\mathsf{MAC}}[\mathsf{AuthKey}, \mathsf{EncKey}, \mathsf{InsecCh}] \\ \subseteq \quad \mathsf{SecChDg}^{\phi_1} \cap \mathsf{SecChDg}^{\phi_2} \cap \mathsf{SecChDg}^{\phi_3},\end{aligned}$$

where ϕ_1 to ϕ_3 formalize the interval-wise relaxations. Another protocol can then simply assume the overly idealized downgradable secure channel SecChDg, with our novel composition theorem taking care of devising the appropriate overall security statement. We formalize this type of relaxation and the corresponding composition theorem in the next section, i.e., Section 5.3.

Translating the approach to another composable framework, such as UC, might be feasible but non-trivial. First, one might try to formalize a single interval-wise guarantee as a different corruption model, where for instance the adversary simply does not get the encryption key to securely realize a functionality analogon to SecChDg. To then compose this step with a SecChDg-hybrid statement, one would probably require some compiler translating the statement. We, thus, believe that formalizing our results in CC that allows for arbitrary specifications is both simpler and more natural.

A remark on adaptive versus static security

Our security statement makes a static case separation on the intervals considered. This might raise the question as to how this differs from simply considering static corruptions only. We would like to stress that our statement is about a real-world system, where the environment gets to adaptively (depending on all the outputs it sees) choose when the appropriate keys are leaked. Hence, our notion lies somewhere in between the traditional notions of static and adaptive security.

To which extent our notion suffices in practice, and when a stronger traditional adaptive statement is required, is in our opinion an interesting open research problem. On the one hand, fully adaptively secure notions, without doubt, play a crucial role as a technical tool in many cryptographic constructions. On the other hand, very few cases are known where the overall security of an application actually seems to be meaningfully impacted by adaptiveness. For instance, consider the folklore example of an MPC protocol where an adversary knows which party she has to corrupt based on some observed value during the execution. Nevertheless, for a polynomially sized adversary structure (i.e., choices which parties to corrupt), the adversary could still guess upfront, implying that even traditional static security would suffice. This is for instance the case if there are only logarithmically (or constant)

many parties overall.

Moreover, even if there super-polynomially many choices, it could still be that our interpretation of the static result is wrong: if we distinguish n static cases, and in each one of them a certain property is violated with probability ϵ, then all we can say is that by the union bound the probability of a property being violated is bounded by $n\epsilon$. Hence, concluding from ϵ being negligible that the protocol is overall secure, might simply not be sound in the first place.

A remark on stronger security guarantees

The primary goal of this work is to express the security guarantee of certain schemes in a composable framework, for which so far this has not been possible. This does not contradict stronger security notions, such as non-committing encryption, being of use as well. For instance, insisting that the simulator can explain the ciphertexts (in the traditional notion) formalizes that the ciphertexts are never of any value—in a broader sense than confidentiality. This might play an important role in advanced properties such as deniability, or e.g. in a scenario where an adversary wants to prove to another party that he managed to wiretap the channel before the transmitted message and the corresponding encryption key are publicly announced. Phrasing that no adversary can succeed requires the simulator to work beyond the public announcement, and achieving it requires non-committing encryption. Otherwise, committing to the ciphertext ahead of the public announcement should convince the other party.

5.3 Interval-Wise Guarantees: Definitions

In this section, we formalize interval-wise guarantees as a type of relaxation and provide the corresponding composition theorem. In the spirit of modularity, we proceed in several steps. First, we introduce one relaxation that waives all guarantees after a certain point, and second, the complementary one that waives all guarantees before a certain event. Third, we combine those relaxations and show that it fits well into the existing theory. Finally, we present the resulting construction notion and phrase the motivating example therein.

5.3.1 Guarantees up to Some Point

As we have seen in the motivational example, the confidentiality of the messages should be guaranteed *until* the key is leaked. To phrase this, we, on a high level, only require that the simulator works up to this event. We formalize this as a novel type of relaxation consisting of all systems behaving equally up to this point. To this end, for a resource R, we consider the modified resource that halts once a certain predicate on the global event history is satisfied.

Definition 5.3.1. Let R denote a resource, and let $P(\mathcal{E})$ denote a monotone predicate on the global event history. That is, if $\mathcal{E}$ is a prefix of $\mathcal{E}'$ then $P(\mathcal{E}) \to P(\mathcal{E}')$. Then, we denote by $\mathrm{until}_P(\mathsf{R})$ the resource that behaves like R but *halts* the moment $P(\mathcal{E})$ becomes true. That is, it no longer triggers any further events and all subsequent (including the one for the query that triggered the condition) answers are the special symbol $\perp$.

Getting back to our example, consider the resource $\mathrm{until}_P(\mathsf{SecChDg})$ for $P(\mathcal{E}) \coloneqq \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}$, depicted in Figure 5.2. Since this resource no longer produces any output once either event occurred, it clearly never leaks the messages to Eve and removes Eve's capability of injecting messages. Hence, the resulting resource closely matches the expected secure channel when ignoring key exposures.

We now define the according relaxation, which maps a system to the set of all systems that behave equivalently up to some event.

Definition 5.3.2. Let P be a monotone predicate on the global event history, indicating until when the behavior must be the same as the one of the resource R. Then, the induced relaxation on a resource R, denoted $\mathsf{R}^{P]}$, is defined as

$$\mathsf{R}^{P]} \coloneqq \left\{ \mathsf{S} \;\middle|\; \mathrm{until}_P(\mathsf{R}) = \mathrm{until}_P(\mathsf{S}) \right\}$$

We call such a relaxation generally an *until-relaxation*.

As with the ϵ-relaxation, the statements only become reusable and thus truly composable if we understand how the until-relaxation interacts with the other elements of the framework. For this, first observe that equality up to some point is monotone, i.e., if two resources are equivalent

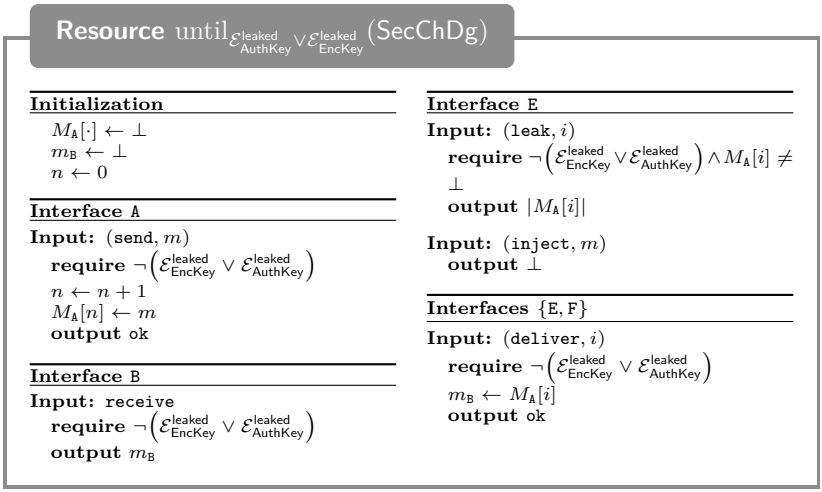


Figure 5.2: The secure channel from Figure 5.1 when halted once either key leaks. In contrast to the original one, this resource never leaks the actual messages.

up to some point, they are also equivalent up to every earlier point. This furthermore implies that two until-relaxations add up in the natural manner, as follows.

Theorem 5.3.3. *Let* R *and* S *be two resources, and let* P_1 *and* P_2 *be two monotone predicates. Then, we have*

$$\text{until}_{P_1}(\mathsf{R}) = \text{until}_{P_1}(\mathsf{S}) \implies \text{until}_{P_1 \vee P_2}(\mathsf{R}) = \text{until}_{P_1 \vee P_2}(\mathsf{S}).$$

In particular, for every specification $\mathcal{R}$*, we have* $\mathcal{R}^{P_1]} \subseteq \left(\mathcal{R}^{P_1]}\right)^{P_2]} \subseteq \mathcal{R}^{P_1 \vee P_2]}$.

Proof. The first property follows directly from the definition of the until()-projection. To prove the second property, let $\mathsf{S} \in \left(\mathcal{R}^{P_1]}\right)^{P_2]}$. Then, there exists $\mathsf{T} \in \mathcal{R}^{P_1]}$ such that $\text{until}_{P_2}(\mathsf{S}) = \text{until}_{P_2}(\mathsf{T})$. Moreover, there exists $\mathsf{R} \in \mathcal{R}$ such that $\text{until}_{P_1}(\mathsf{T}) = \text{until}_{P1}(\mathsf{R})$. By the first property we, thus, obtain $\text{until}_{P_1 \vee P_2}(\mathsf{S}) = \text{until}_{P_1 \vee P_2}(\mathsf{T}) = \text{until}_{P_1 \vee P_2}(\mathsf{R})$, concluding the proof. □

Furthermore, on a positive note, the relaxation is compatible with both protocol application and parallel composition, as expressed by the following theorem. Those compatibility properties—analogously to Corollary 2.2.13—also directly imply sequential and parallel composition properties. For the lack of use, we however omit explicitly stating them.

Theorem 5.3.4. *The until-relaxation is compatible with protocol attachment, i.e.,* $\boldsymbol{\pi}\big(\mathcal{R}^{P]}\big) \subseteq \big(\boldsymbol{\pi}\mathcal{R}\big)^{P]}$ *and with parallel composition, i.e.,* $\big[\mathcal{R}^{P]}, \mathcal{S}\big] \subseteq \big[\mathcal{R}, \mathcal{S}\big]^{P]}$.

Proof. For the first property, consider an arbitrary element of $\boldsymbol{\pi}\big(\mathcal{R}^{P]}\big)$, i.e., $\boldsymbol{\pi}\mathsf{T}$ for an arbitrary $\mathsf{T} \in \mathcal{R}^{P]}$. Hence, there must exist a $\mathsf{R} \in \mathcal{R}$ such that $\mathrm{until}_P(\mathsf{R}) = \mathrm{until}_P(\mathsf{T})$. Using that $\mathrm{until}_P(\boldsymbol{\pi}\,\mathrm{until}_P(\mathsf{R})) = \mathrm{until}_P(\boldsymbol{\pi}\mathsf{R})$ for any resource R, implies

$$\mathrm{until}_P(\boldsymbol{\pi}\mathsf{R}) = \mathrm{until}_P(\boldsymbol{\pi}\,\mathrm{until}_P(\mathsf{R})) = \mathrm{until}_P(\boldsymbol{\pi}\,\mathrm{until}_P(\mathsf{T})) = \mathrm{until}_P(\boldsymbol{\pi}\mathsf{T})$$

and, thus, $\boldsymbol{\pi}\mathsf{T} \in \big(\boldsymbol{\pi}\mathsf{R}\big)^{P]} \subseteq \big(\boldsymbol{\pi}\mathcal{R}\big)^{P]}$. The second property follows analogously observing that $\mathrm{until}_P([\mathrm{until}_P(\mathsf{R}), \mathsf{S}]) = \mathrm{until}_P([\mathsf{R}, \mathsf{S}])$ for all R and S. □

Unfortunately, however, the until-relaxation does not commute directly with the ϵ-relaxation, as expressed by the following theorem.

Theorem 5.3.5. *There exist specifications* $\mathcal{R}$ *and* $\mathcal{S}$*, a monotone predicate* P*, and a function* ϵ *mapping distinguishers to values in* $[0,1]$ *such that*

$$\big(\mathcal{R}^{P]}\big)^{\epsilon} \not\subseteq \big(\mathcal{R}^{\epsilon}\big)^{P]} \qquad \text{and} \qquad \big(\mathcal{S}^{\epsilon}\big)^{P]} \not\subseteq \big(\mathcal{S}^{P]}\big)^{\epsilon}.$$

Proof (Sketch). In the following, let R denote the following resource that provides two interfaces: R initially chooses a seed $s \in \{0,1\}^m$ uniformly at random. Upon a trigger input at the first interface it outputs $\mathrm{PRG}(s)$ (for some m-bits to n-bits PRG). Upon a trigger input at the second interface, it triggers a $\mathcal{E}^{\mathsf{leaked}}$ event and outputs s. Let the resource S work analogously except that it outputs a uniformly distributed an independent value $t \in \{0,1\}^n$ instead of the PRG output. Furthermore, let $P(\mathcal{E}) \coloneqq \mathcal{E}^{\mathsf{leaked}}$ and let ϵ denote the reduction to the security of the PRG.

Using $\mathcal{R} \coloneqq \{\mathsf{R}\}$ it is now easy to see that $\mathsf{S} \in (\mathcal{R}^{P]})^{\epsilon}$: Consider a hybrid system T that outputs $\mathrm{PRG}(s')$ and s for independent and s

and s' u.a.r. Then, we have $\mathsf{T} \in \mathcal{R}^{P]}$, since the first output is equally distributed and the second blinded. Moreover, distinguishing T from S boils down to breaking the PRG-security, and thus $\mathsf{S} \in \mathsf{T}^{\epsilon}$. On the other hand, one can show that $\mathsf{S} \notin \left(\mathcal{R}^{\epsilon}\right)^{P]}$. To see this, observe that if the PRG is secure, then $\mathcal{R}^{\epsilon} \approx \mathcal{R}$. Especially, leaking the seed prevents us from replacing the first output by a truly random string. Thus, applying the until-relaxation still does not contain S, concluding the first part of the proof. For the other direction consider $\mathcal{S} \coloneqq \{\mathsf{S}\}$. Using an analogous argument one can show that $\mathsf{R} \in \left(\mathcal{S}^{\epsilon}\right)^{P]}$ but $\mathsf{R} \notin \left(\mathcal{S}^{P]}\right)^{\epsilon}$. □

This not only raises the question which order actually corresponds to the intuitive interpretation of such a combination—the set of all systems which behave equally until the condition is triggered assuming the assumption of ϵ is valid—but also restricts reuse of such statement. That is, if one construction assumes $\mathcal{S}^{P]}$ to obtain $\mathcal{T}$, and another one constructs $\mathcal{S}^{\epsilon}$ instead, then adjusting the former construction to assume $\mathcal{S}^{\epsilon}$ instead is non-trivial. As a consequence, we will introduce a combined relaxation in Section 5.3.3, resolving both issues.

5.3.2 Guarantees From Some Point On

In this section, we now consider the complementing type of guarantees: guarantees that only hold from a certain point on. Formalizing such guarantees in a model where an adaptive environment interacts with the resource is, however, quite delicate. In this work, we thus opt for a rather simple (and restricted) version of it, where we use again a monotone condition on the global event history. We then define the projection that disables access to a system R before that condition is met. Clearly, the condition must rely on "external" events only (the ones not controlled by R), i.e., satisfying it must not require accessing the resource itself.

Definition 5.3.6. Let $P(\mathcal{E})$ denote a monotone predicate on the global event history. For a resource R, let $\mathrm{from}_P(\mathsf{R})$ denote the resource that behaves like R, except that it only accepts queries once $P(\mathcal{E})$ is true (and before only returns $\perp$).

For instance, the resource $\mathrm{from}_{\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}}(\mathsf{SecChDg})$ only answers queries once the environment triggered the event $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}$. Thus, in contrast to

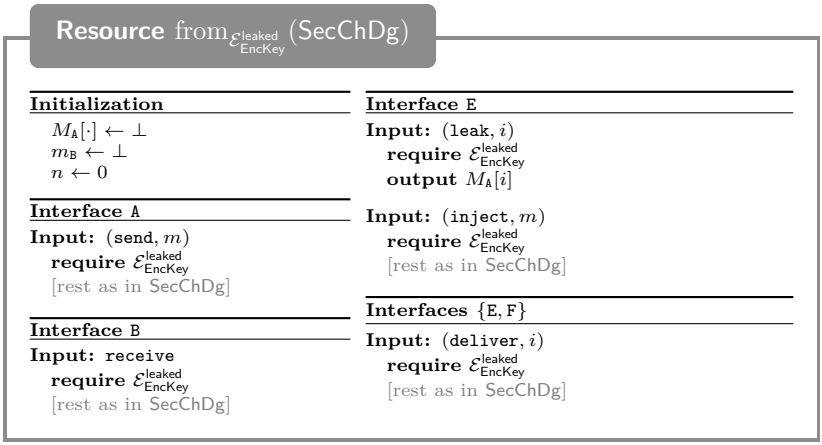


Figure 5.3: The secure channel from Figure 5.1 that it only accepts queries upon leakage of the encryption key. Note that this resource does not provide confidentiality.

SecChDg, this resource always leaks the full message of the adversary, in line with our intuition that it describes the behavior after the key has been exposed. A formal definition of the resource is depicted in Figure 5.3.

Based on this projection, we now introduce the corresponding relaxation.

Definition 5.3.7. Let $P(\mathcal{E})$ be a monotone predicate, indicating from which point on the behavior must be the same as the one of the resource R. Then, the induced relaxation on a resource R, denoted $\mathsf{R}^{[P}$, is defined as

$$\mathsf{R}^{[P} \coloneqq \left\{ \mathsf{S} \;\middle|\; \text{from}_P(\mathsf{R}) = \text{from}_P(\mathsf{S}) \right\}$$

We call such a relaxation generally a *from-relaxation*.

The way the from-relaxation interacts with the other elements of the theory is analogous to the until-relaxation. First, two from-relaxations add up naturally: if we relax the guarantees offered by a specification to only hold from the moment P_1 is satisfied, and then further relax

them to only hold once P_2 is satisfied, then the guarantees only hold once $P_1 \wedge P_2$ is satisfied.

Theorem 5.3.8. *Let* R *and* S *be two resources, and let* P_1 *and* P_2 *be monotone predicates on the global event history. Then, we have*

$$\mathrm{from}_{P_1}(\mathsf{R}) = \mathrm{from}_{P_1}(\mathsf{S}) \implies \mathrm{from}_{P_1 \wedge P_2}(\mathsf{R}) = \mathrm{from}_{P_1 \wedge P_2}(\mathsf{S}).$$

In particular, for every specification $\mathcal{R}$*, we have* $\mathcal{R}^{[P_1} \subseteq \left(\mathcal{R}^{[P_1}\right)^{[P_2} \subseteq \mathcal{R}^{[P_1 \wedge P_2}$*.*

Proof. The proof is analogous to the one of Theorem 5.3.3.

Second, the relaxation is compatible with protocol application and parallel composition, which moreover implies that it graciously interacts with the basic construction notion.

Theorem 5.3.9. *The from-relaxation is compatible with protocol application, i.e.,* $\boldsymbol{\pi}\left(\mathcal{R}^{[P}\right) \subseteq \left(\boldsymbol{\pi}\mathcal{R}\right)^{[P}$ *and with parallel composition, i.e.,* $\left[\mathcal{R}^{[P}, \mathcal{S}\right] \subseteq \left[\mathcal{R}, \mathcal{S}\right]^{[P}$*.*

Proof. The proof is analogous to the one of Theorem 5.3.4.

Analogously to the until-relaxation, the from-relaxation, however, does not commute with the ϵ-relaxation.

Theorem 5.3.10. *There exist specifications* $\mathcal{R}$ *and* $\mathcal{S}$*, a monotone predicate* $P(\mathcal{E})$*, and a function* ϵ *mapping distinguishers to values in* $[0,1]$ *such that*

$$\left(\mathcal{R}^{[P}\right)^{\epsilon} \not\subseteq \left(\mathcal{R}^{\epsilon}\right)^{[P} \qquad \text{and} \qquad \left(\mathcal{S}^{\epsilon}\right)^{[P} \not\subseteq \left(\mathcal{S}^{[P}\right)^{\epsilon}.$$

Proof (Sketch). The counter example works similarly to the one from Theorem 5.3.5. Consider a resource R with two interfaces that initially chooses a PRG-seed s uniformly at random. At the first interface it outputs $\mathrm{PRG}(s)$. At the second interface it outputs s, but only before some event $\mathcal{E}^{\mathsf{secure}}$. The second resource S works analogously except that it outputs an independent uniform random string at the first interface instead of the PRG output. Let now $\mathcal{R} \coloneqq \{\mathsf{R}\}$, $\mathcal{S} \coloneqq \{\mathsf{S}\}$, and $P \coloneqq \{\mathcal{E}^{\mathsf{secure}}\}$. Analogous to the proof from Theorem 5.3.5, one can now show that $\mathsf{S} \in \left(\mathcal{R}^{[P}\right)^{\epsilon}$ but $\mathsf{S} \notin \left(\mathcal{R}^{\epsilon}\right)^{[P}$, and furthermore that $\mathsf{R} \in \left(\mathcal{S}^{\epsilon}\right)^{[P}$ but $\mathsf{S} \notin \left(\mathcal{S}^{[P}\right)^{\epsilon}$. □

Finally, consider the interaction between the from- and the until-relaxation. While the from-projection and the until-projection commute, i.e.,

$$\mathrm{from}_{P_1}(\mathrm{until}_{P_2}(\mathsf{R})) = \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{R})),$$

it is an interesting open question whether the two respective relaxations actually commute. As a consequence, we introduce a combined from-until relaxation in the next subsection.

5.3.3 The Interval-Wise Relaxation

As we have seen, the ϵ-relaxation commutes neither with the until-relaxation nor the from-relaxation, and its unclear whether the from- and until-relaxations do. This impedes modularity and reusability of the statements. Furthermore, it also deteriorates the intuitive semantics of the statements: if for instance we want to express that a system behaves like a certain ideal up to some point, and under certain computational assumptions, which order of the relaxations is the right one and should be proven? To alleviate those issues, in this section, we introduce two combined relaxations that build on the atomic ones introduced in the previous section. We then show that they both have natural semantics and clean properties.

First, we consider a relaxation that combines the from- and until-relaxation, thereby alleviating the issue that those relaxations might not commute.

Definition 5.3.11. Let $P_1(\mathcal{E})$ and $P_2(\mathcal{E})$ be two monotone predicates, indicating from when until when the resource must behave like R. We then define the following relaxation

$$\mathsf{R}^{[P_1,P_2]} \coloneqq \left\{\mathsf{S} \;\middle|\; \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{R})) = \mathrm{until}_{P_2}(\mathrm{from}_{P_1}(\mathsf{S}))\right\}.$$

While this combined relaxation apparently neither corresponds to $\left(\mathsf{R}^{[P_1}\right)^{P_2]}$ nor $\left(\mathsf{R}^{P_2]}\right)^{[P_1}$, it interestingly corresponds to the transitive closure thereof. Taking the transitive closure, moreover, also restores symmetry, i.e., $\mathsf{S} \in \mathsf{R}^{[P_1,P_2]} \Leftrightarrow \mathsf{R} \in \mathsf{S}^{[P_1,P_2]}$, lost by each of the two individual combinations. Overall, this indicates that the combined relaxation best corresponds to the intuition of the "almost-as-good" relation it should intuitively represent.

Theorem 5.3.12. *For any resource* R *and any monotone predicates* P_1 *and* P_2*, we have*

$$\begin{aligned}\mathsf{R}^{[P_1,P_2]} &= \bigcup_{n\in\mathbb{N}} \Big(\bigcup \big\{ \mathsf{R}^{\phi_1\cdot\phi_2\cdots\phi_n} \mid \forall i \leq n : \phi_i \in \{P_2],[P_1\}\big\}\Big) \\ &= \Big(\big(\mathsf{R}^{[P_1}\big)^{P_2]}\Big)^{[P_1} = \Big(\big(\mathsf{R}^{P_2]}\big)^{[P_1}\Big)^{P_2]},\end{aligned}$$

where $\mathsf{R}^{\phi_1\cdot\phi_2\cdots\phi_n}$ *is a shorthand notation for first applying* ϕ_1*, then* ϕ_2*, until* ϕ_n*.*

Proof. The proof can be found in Appendix B.1.1.

We can now leverage this alternative definition to directly derive properties about the combined relaxations based on the proven properties of the two underlying ones. In particular, we can show that two such relaxations add up in the expected manner and are compatible with both protocol application as well as parallel composition.

Theorem 5.3.13. *For every specification* $\mathcal{R}$*, and all monotone predicates* P_1*,* P_2*,* P_1'*, and* P_2'*, we have* $\big(\mathcal{R}^{[P_1,P_2]}\big)^{[P_1',P_2']} \subseteq \mathcal{R}^{[P_1\wedge P_1',P_2\vee P_2']}$*.*

Proof. This follows directly by appropriately combining Theorems 5.3.3, 5.3.8 and 5.3.12. □

Theorem 5.3.14. *The combined relaxation is both compatible with protocol application, i.e.,* $\boldsymbol{\pi}\big(\mathcal{R}^{[P_1,P_2]}\big) \subseteq \big(\boldsymbol{\pi}\mathcal{R}\big)^{[P_1,P_2]}$ *and with parallel composition, i.e.,* $\big[\mathcal{R}^{[P_1,P_2]},\mathcal{S}\big] \subseteq \big[\mathcal{R},\mathcal{S}\big]^{[P_1,P_2]}$*.*

Proof. By Theorem 5.3.12 we have that $\mathcal{R}^{[P_1,P_2]} = \Big(\big(\mathsf{R}^{[P_1}\big)^{P_2]}\Big)^{[P_1}$. Using the compatibility of the from-relaxation and until-relaxation, i.e., Theorems 5.3.4 and 5.3.9, directly implies the desired properties. □

As we have seen, neither the until- nor the from-relaxation commute with the computational ϵ-relaxation, and the same holds true for the from-until-relaxation as well. As a consequence, neither $(\mathsf{R}^{[P_1,P_2]})^{\epsilon}$ nor $(\mathsf{R}^{\epsilon})^{[P_1,P_2]})$ seems to capture the set of all systems that behave like $\mathcal{R}$ in the interval $[P_1,P_2]$ assuming that the computational problem encoded in ϵ is hard. In the spirit of the combined from-until relaxation,

we solve this issue by introducing a combined relation. Since the ϵ relaxation is not idempotent, but the epsilons add up, taking the transitive closure, however, does not match the desired relaxation but the following restricted version of transitive closure does.

Definition 5.3.15. For two monotone predicates P_1 and P_2, and a function ϵ mapping distinguishers to values in $[0,1]$, we define the following relaxation:

$$\mathsf{R}^{[P_1,P_2]:\epsilon} \coloneqq \left(\left(\mathsf{R}^{[P_1,P_2]}\right)^{\epsilon}\right)^{[P_1,P_2]},$$

and call such a relaxation an *interval-wise relaxation.*

We now prove that the interval-wise relaxation has all the desired properties.

Theorem 5.3.16. *Let P_1 and P_2 be two monotone predicates, and let ϵ be a function mapping distinguishers to values in $[0,1]$. Then, for any specification $\mathcal{R}$ we have*

$$\left(\mathcal{R}^{[P_1,P_2]:\epsilon}\right)^{[P_1',P_2']:\epsilon'} \subseteq \mathcal{R}^{[P_1\wedge P_1', P_2\vee P_2']:\epsilon_{[P_1\wedge P_1', P_2\vee P_2']}+\epsilon'_{[P_1\wedge P_1', P_2\vee P_2']}},$$

where $\epsilon_{[P_1\wedge P_1', P_2\vee P_2']}(D) \coloneqq \epsilon(\mathsf{D}\circ \mathrm{until}_{P_2\vee P_2'}\circ \mathrm{from}_{P_1\wedge P_1'})$, i.e., the performance of the distinguisher interacting with the projected resource, and analogously for $\epsilon'_{[P_1\wedge P_1', P_2\vee P_2']}$.

Proof. The proof can be found in Appendix B.1.2.

Theorem 5.3.17. *The interval-wise relaxation is compatible with protocol application, i.e., $\boldsymbol{\pi}\left(\mathcal{R}^{[P_1,P_2]:\epsilon}\right) \subseteq \left(\boldsymbol{\pi}\mathcal{R}\right)^{[P_1,P_2]:\epsilon_{\boldsymbol{\pi}}}$ and with parallel composition, i.e., $\left[\mathcal{R}^{[P_1,P_2]:\epsilon}, \mathcal{S}\right] \subseteq \left[\mathcal{R},\mathcal{S}\right]^{[P_1,P_2]:\epsilon_{\mathcal{S}}}$.*

Proof. By definition we have $\mathsf{R}^{[P_1,P_2]:\epsilon} \coloneqq \left(\left(\mathsf{R}^{[P_1,P_2]}\right)^{\epsilon}\right)^{[P_1,P_2]}$. Using the compatibility of the ϵ-relaxation and the from-until-relaxation, i.e., Theorems 2.2.11 and 5.3.14, directly implies the result. □

5.3.4 The Resulting Construction Notion

Based on the interval-wise relaxation, we now introduce our new construction notion. For a single interval, we then consider statements of the following type.

Definition 5.3.18. Let $\mathcal{R}$ and $\mathcal{S}$ be specifications, let $\boldsymbol{\pi}$ be a protocol for $\mathcal{R}$, let $\boldsymbol{\sigma}$ be a simulator for $\mathcal{S}$, and let ϵ be a function that maps distinguishers to a value in $[0,1]$. Moreover, let P_1 and P_2 be monotone predicates on the global event history. Then, we define

$$\mathcal{R} \xrightarrow[\mathsf{interval}]{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon,[P_1,P_2]} \mathcal{S} \;:\Longleftrightarrow\; \boldsymbol{\pi}\mathcal{R} \subseteq (\boldsymbol{\sigma}\mathcal{S})^{[P_1,P_2]:\epsilon},$$

and say that the protocol $\boldsymbol{\pi}$ constructs $\mathcal{S}$ from $\mathcal{R}$ within the interval $[P_1,P_2]$ and with respect to error ϵ and the simulator $\boldsymbol{\sigma}$.

When considering multiple intervals, i.e., asserting that the real-world specification is a subset of the intersection of multiple such statements, we can simply express this as the conjunction of the appropriate construction statements.

For ease of comparing with asymptotic game-based notions, we however do introduce an asymptotic variant thereof.

Definition 5.3.19. Let $\mathcal{R} \subseteq \Theta_{poly}$ and $\mathcal{S} \subseteq \Theta_{poly}$ be two specifications of efficient resource families, and let $\langle \boldsymbol{\pi}_\lambda \rangle_{\lambda\in\mathbb{N}}$ be an efficient protocol family. Moreover, let P_1 and P_2 be monotone predicates on the global event history. If there exists an efficient simulator family $\langle \boldsymbol{\sigma}_\lambda \rangle_{\lambda\in\mathbb{N}}$, and a negligible family of functions $\langle \epsilon_\lambda \rangle_{\lambda\in\mathbb{N}}$, such that

$$\forall \lambda \in \mathbb{N} : \boldsymbol{\pi}_\lambda \mathcal{R}_\lambda \subseteq (\boldsymbol{\sigma}_\lambda \mathcal{S}_\lambda)^{[P_1,P_2]:\epsilon_\lambda},$$

and say that the protocol $\boldsymbol{\pi}$ asymptotically constructs $\mathcal{S}$ from $\mathcal{R}$ within the interval $[P_1,P_2]$, and denote it by

$$\mathcal{R} \xrightarrow[\mathsf{interval\text{-}asym}]{\langle \boldsymbol{\pi}_\lambda \rangle_{\lambda\in\mathbb{N}},[P_1,P_2]} \mathcal{S}.$$

Note that this construction notion subsumes all the ones introduced in Section 2.2. In particular, instantiating $P_1 = \mathtt{true}$, $P_2 = \mathtt{false}$, $\epsilon(\mathsf{D}) = 0$, and $\boldsymbol{\sigma} = \mathsf{id}$, i.e., the identity converter, yields $(\mathsf{id}\mathcal{S})^{[\mathtt{true},\mathtt{false}]:0} = \mathcal{S}$.

Application to the running example

In our example, we want to phrase that the symmetric encryption protocol constructs the secure channel from the authenticated one and the key in the corresponding intervals.

Proposition 5.3.20. *Let $\boldsymbol{\pi}_{\mathsf{ENC}} = (\pi_{\mathsf{enc}}, \pi_{\mathsf{dec}})$ denote the protocol securing communication using a symmetric encryption scheme. Then, for the resources in Figure 5.1, there exist (efficient) simulators $\boldsymbol{\sigma_1}$, $\boldsymbol{\sigma_2}$, and $\boldsymbol{\sigma_3}$ such that*

$$\bigwedge_{(\boldsymbol{\sigma},\epsilon,P_1,P_2)\in\Omega} \Big(\big[\mathsf{EncKey}, \mathsf{AuthChDg}\big] \xrightarrow[\mathsf{interval}]{\boldsymbol{\pi}_{\mathsf{ENC}},\boldsymbol{\sigma},\epsilon,[P_1,P_2]} \mathsf{SecChDg} \Big)$$

for

$$\Omega \coloneqq \big\{ \big(\boldsymbol{\sigma}_1, \epsilon_{\mathsf{CPA}}, \mathtt{true}, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}\big), \big(\boldsymbol{\sigma}_2, 0, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}, \mathtt{false}\big), \big(\boldsymbol{\sigma}_3, 0, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}, \mathtt{false}\big) \big\},$$

where ϵ_{CPA} denotes a simple reduction from distinguishing the secure and authenticated channel (without key leakage) to the IND-CPA game.

Proof. A proof sketch is presented in Appendix B.1.3.

Composition

Finally, we finish the section by stating the composition guarantees of this type of construction statement. It follows directly from the properties proven about the interval-wise relaxation in Theorems 5.3.16 and 5.3.17.

Theorem 5.3.21. *Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be specifications, let $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ be protocols, let $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ be simulators, and let ϵ and ϵ' be functions mapping distinguishers to a value in $[0,1]$. Moreover, let P_1, P_2, P_1', P_2' be monotone predicates on the event history. Then, we have*

$$\mathcal{R} \xrightarrow[\mathsf{interval}]{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon,[P_1,P_2]} \mathcal{S} \quad \wedge \quad \mathcal{S} \xrightarrow[\mathsf{interval}]{\boldsymbol{\pi}',\boldsymbol{\sigma}',\epsilon',[P_1',P_2']} \mathcal{T} \implies \mathcal{R} \xrightarrow[\mathsf{interval}]{\boldsymbol{\pi}'\circ\boldsymbol{\pi},\boldsymbol{\sigma}\circ\boldsymbol{\sigma}',\tilde{\epsilon},[P_1\wedge P_1',P_2\vee P_2']} \mathcal{T},$$

where $\tilde{\epsilon} := (\epsilon_{\boldsymbol{\pi}'})_{[P_1 \wedge P_1', P_2 \vee P_2']} + (\epsilon'_{\boldsymbol{\sigma}})_{[P_1 \wedge P_1', P_2 \vee P_2']}$. *Furthermore, we have*

$$\mathcal{R} \overset{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon,[P_1,P_2]}{\underset{\mathsf{interval}}{\longmapsto}} \mathcal{S} \quad \Longrightarrow \quad [\mathcal{R},\mathcal{T}] \overset{\boldsymbol{\pi},\boldsymbol{\sigma},\epsilon_{\mathcal{T}},[P_1,P_2]}{\underset{\mathsf{interval}}{\longmapsto}} [\mathcal{S},\mathcal{T}].$$

Proof. We first prove sequential composition. From the first part of the assumption, Theorem 5.3.17, and composition order invariance we obtain

$$\boldsymbol{\pi}'\boldsymbol{\pi}\mathcal{R} \subseteq \boldsymbol{\pi}'\big((\boldsymbol{\sigma}\mathcal{S})^{[P_1,P_2]:\epsilon}\big) \subseteq (\boldsymbol{\pi}'\boldsymbol{\sigma}\mathcal{S})^{[P_1,P_2]:\epsilon_{\boldsymbol{\pi}'}} \subseteq (\boldsymbol{\sigma}\boldsymbol{\pi}'\mathcal{S})^{[P_1,P_2]:\epsilon_{\boldsymbol{\pi}'}}.$$

Moreover, using the second part of the assumption and Theorem 5.3.17 yields

$$\boldsymbol{\sigma}\boldsymbol{\pi}'\mathcal{S} \subseteq \boldsymbol{\sigma}\big((\boldsymbol{\sigma}'\mathcal{T})^{[P_1',P_2']:\epsilon'}\big) \subseteq (\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1',P_2']:\epsilon'_{\boldsymbol{\sigma}}}.$$

Combining the two statements, and using the monotonicity of relaxations, we get

$$\boldsymbol{\pi}'\boldsymbol{\pi}\mathcal{R} \subseteq \big((\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1',P_2']:\epsilon'_{\boldsymbol{\sigma}}}\big)^{[P_1,P_2]:\epsilon_{\boldsymbol{\pi}'}} \subseteq (\boldsymbol{\sigma}\boldsymbol{\sigma}'\mathcal{T})^{[P_1 \wedge P_1', P_2 \vee P_2']:\tilde{\epsilon}},$$

where in the last step we used Theorem 5.3.16, directly implying the sequential composition property. Now consider parallel composition. We directly obtain

$$\begin{aligned}
\boldsymbol{\pi}\big[\mathcal{R},\mathcal{T}\big] &= \big[\boldsymbol{\pi}\mathcal{R},\mathcal{T}\big] \\
&\subseteq \big[(\boldsymbol{\sigma}\mathcal{S})^{[P_1,P_2]:\epsilon},\mathcal{T}\big] \\
&\subseteq \big[\boldsymbol{\sigma}\mathcal{S},\mathcal{T}\big]^{[P_1,P_2]:\epsilon_{\mathcal{T}}} \\
&= \big(\boldsymbol{\sigma}\big[\mathcal{S},\mathcal{T}\big]\big)^{[P_1,P_2]:\epsilon_{\mathcal{T}}},
\end{aligned}$$

where in the first step we used composition order invariance, in the second step the assumption, in the third step Theorem 5.3.17, and in the last step composition order invariance again. □

Note that since the interval-wise construction notion subsumes the plain one, the above composition theorem also allows to combine the respective constructions. For instance, in our example, we can compose the construction of $\mathsf{AuthChDg}$ from Proposition 5.2.1 (according to the standard notion) with the interval-wise construction of $\mathsf{SecChDg}$ from Proposition 5.3.20.

5.4 Application to Commitment Schemes and Coin-Tossing

In this section, we present a composable formalization of (information-theoretically binding) commitments that can be constructed in the plain model. To this end, we formalize the properties of commitment schemes—correctness, binding, and hiding—each as individual specifications. Thereby, hiding is formalized using the interval-wise guarantees introduced in the previous section. We then apply Blum's coin-tossing protocol on top of it. While, obviously, the resulting specifications are not sufficient to be used as a CRS, we show that it is unbiased.

5.4.1 Perfectly Binding Commitments

While UC commitments [CF01] provide clean and strong guarantees, unfortunately they intrinsically require setup assumptions such as a common reference string. Nevertheless, for many protocols, regular commitments only satisfying the classical game-based properties seem to suffice. This raises the question: can we formalize a weaker yet composable security notion for (non-interactive) commitments?

In Constructive Cryptography, the security of a commitment scheme is phrased using three different constructions [MR11], for each set of potentially dishonest parties (ignoring the case of both parties being dishonest). Typically, this is presented as one construction parametrized in the set of honest parties, where the ideal specification consists of a filtered resource. That is, for each party P, a filter ϕ_{P} is specified that when connected to the resource limits the honest party's capabilities. However, there is no fundamental reason for those three construction statements' specifications to be of some unified type. As a result, we henceforth focus on specifying each property—hiding, binding, and correctness—individually, altough centered around the same commitment resource depicted in Figure 5.4.

We start with formalizing correctness.

Definition 5.4.1. Let $\boldsymbol{\pi}_{\mathsf{com}} = (\pi^{\mathtt{A}}_{\mathsf{com}}, \pi^{\mathtt{B}}_{\mathsf{com}})$ denote a non-interactive commitment protocol where $\mathtt{A}$ commits a value $m \in \mathcal{M}$ towards $\mathtt{B}$. The

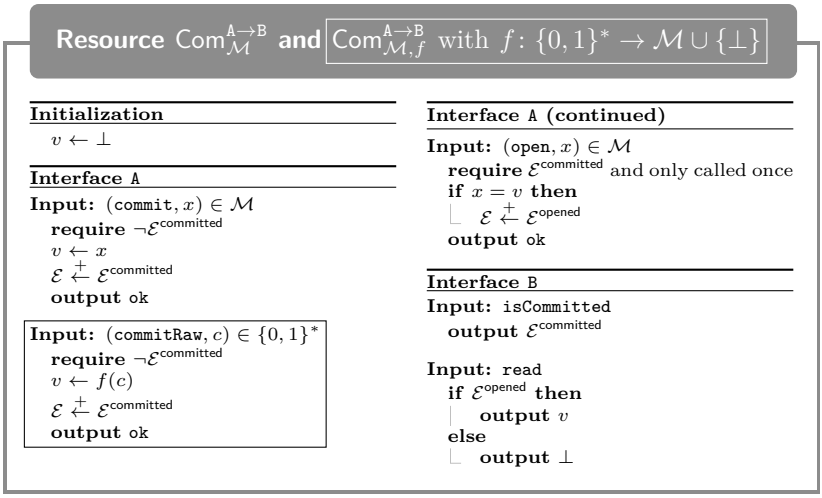


Figure 5.4: The commitment resources for message space $\mathcal{M}$. In the basic version, Alice has to specify the value at the time of commitment, whereas in the unfiltered version she additionally has the ability to commit to $f(c)$.

scheme is said to be (perfectly) *correct* if

$$\left[\mathsf{Ch}_1^{\mathtt{A}\to\mathtt{B}}, \mathsf{Ch}_2^{\mathtt{A}\to\mathtt{B}}\right] \vdash^{\boldsymbol{\pi}_{\mathsf{com}}} \mathsf{Com}_{\mathcal{M}}^{\mathtt{A}\to\mathtt{B}},$$

where $\mathsf{Com}_{\mathcal{M}}^{\mathtt{A}\to\mathtt{B}}$ denotes the commitment resource defined in Figure 5.4, and $\mathsf{Ch}_1^{\mathtt{A}\to\mathtt{B}}$ and $\mathsf{Ch}_2^{\mathtt{A}\to\mathtt{B}}$ denote two single-message communications channels from A to B.

Now we proceed to formalize the hiding property. On an intuitive level, (computational) hiding of a non-interactive commitment scheme requires that the commitment string must not reveal any information about the committed value to the receiver B, until the commitment is opened. Clearly, we can directly apply our notion from Section 5.3 and formalize this using an interval-wise relaxation.

Definition 5.4.2. Let $\boldsymbol{\pi}_{\mathsf{com}} = (\pi^{\mathtt{A}}_{\mathsf{com}}, \pi^{\mathtt{B}}_{\mathsf{com}})$ denote a non-interactive commitment protocol. Then, the scheme is said to be (computationally)

hiding if

$$\left[\mathsf{Ch}_1^{\mathtt{A}\to\mathtt{B}}, \mathsf{Ch}_2^{\mathtt{A}\to\mathtt{B}}\right] \xrightarrow[\mathsf{interval}]{\pi_{\mathsf{com}}^{\mathtt{A}}, \sigma_{\mathsf{com}}^{\mathtt{B}}, \epsilon, [\mathtt{true}, \mathcal{E}^{\mathsf{opened}}]} \mathsf{Com}_{\mathcal{M}}^{\mathtt{A}\to\mathtt{B}},$$

for some simulator $\sigma_{\mathsf{com}}^{\mathtt{B}}$ (attaching to Bob's interfaces only) and some computational assumption encoded in ϵ.

The situation is more challenging with binding. The UC formalization, and analogously $\mathsf{Com}_{\mathcal{M}}^{\mathtt{A}\to\mathtt{B}}$, requires that the adversary inputs the value to which it commits to in the initial phase, in order to formalize that it then cannot be altered anymore. This, however implies that the simulator must be able to extract the value from the commitment string, fundamentally contradicting the hiding property in the plain model. Since such a formalization is just one (albeit convenient) manner to specify that the value is fixed at the end of the commitment phase, we circumvent this impossibility in another manner. To this end, we consider perfect (or information-theoretically secure) commitments only, where the commitment string uniquely determines the committed value. We leverage this considering a resource $\mathsf{Com}_{\mathcal{M},f}^{\mathtt{A}\to\mathtt{B}}$, depicted in Figure 5.4 as well, which allows the dishonest $\mathtt{A}$ to input an arbitrary string x in order to commit to the value $v = f(x)$. Here, $f\colon \{0,1\}^* \to \mathcal{M} \cup \{\bot\}$ denotes an arbitrary function that maps the commitment string either to a message m, or to $\bot$ indicating that it is malformed.

Definition 5.4.3. Let $\boldsymbol{\pi}_{\mathsf{com}} = (\pi_{\mathsf{com}}^{\mathtt{A}}, \pi_{\mathsf{com}}^{\mathtt{B}})$ denote a non-interactive commitment protocol where $\mathtt{A}$ commits a value $m \in \mathcal{M}$ towards $\mathtt{B}$. Then, the scheme is said to be *perfectly binding* if there exists an efficient simulator $\sigma_{\mathsf{com}}^{\mathtt{A}}$ (connecting to Alice's interface only) such that

$$\left[\mathsf{Ch}_1^{\mathtt{A}\to\mathtt{B}}, \mathsf{Ch}_2^{\mathtt{A}\to\mathtt{B}}\right] \xrightarrow[\mathsf{sim}]{\pi_{\mathsf{com}}^{\mathtt{B}}, \sigma_{\mathsf{com}}^{\mathtt{A}}, 0} \left\{\mathsf{Com}_{\mathcal{M},f}^{\mathtt{A}\to\mathtt{B}} \;\middle|\; f\colon \{0,1\}^* \to \mathcal{M} \cup \{\bot\}\right\},$$

where $\mathsf{Com}_{\mathcal{M},f}^{\mathtt{A}\to\mathtt{B}}$ denotes the extended commitment resource defined in Figure 5.4.

As a side note, note that the resource $\mathsf{Com}_{\mathcal{M}}^{\mathtt{A}\to\mathtt{B}}$ can trivially be expressed as a filtered version of $\mathsf{Com}_{\mathcal{M},f}^{\mathtt{A}\to\mathtt{B}}$, where the filter $\phi_{\mathtt{A}}$ removes access to the `commitRaw` oracle. That is, we obviously have $\phi_{\mathtt{A}}\mathsf{Com}_{\mathcal{M},f}^{\mathtt{A}\to\mathtt{B}} = \mathsf{Com}_{\mathcal{M}}^{\mathtt{A}\to\mathtt{B}}$ for every function f.

Remark. Observe that the function f is not necessary efficiently computable. Actually, for a hiding scheme, f cannot be efficiently computable. This, however, does not imply that the overall specification, with the simulator attached, has to contain resources that are not efficiently implementable, as the real-world specification is efficient. Thus, one could always intersect the obtained specification with the specification of all efficient systems, expressing that there is not more computing power that we started of in the real-world. This is somewhat reminiscent of the solution proposed by Broadnax et al. [BDH+17] to deal with inefficient simulators in a manner that retains the expected composition guarantees. Since such a restriction to overall efficiently implementable systems, in our framework, can be treated as a separate statement, we did not make it explicit and focused only on the security properties.

ElGamal commitments

We briefly consider a variant of ElGamal commitments as a concrete instantiation of the above formalized notion. Let $\mathbb{G} = \langle g \rangle$ denote a cyclic group of order n with generator g.

- To commit to a message $m \in \mathbb{G}$, $((g^a, g^b, m \cdot g^{ab}), (a, b)) \leftarrow \mathrm{Commit}(m)$ for $a, b \in \mathbb{Z}_n$ uniformly at random. That is, the commitment string is $(g^a, g^b, m \cdot g^{ab})$ and the opening value (a, b).
- $\mathrm{Open}((c, A, B), (a, b)) \coloneqq c \cdot g^{-ab}$ if $A = g^a$ and $B = g^b$, and $\perp$ otherwise.

Proposition 5.4.4. *Let $\boldsymbol{\pi}_{\mathsf{ElG\text{-}com}}$ denote the pair of converters implementing the aforementioned ElGamal commitment scheme (cf. Appendix B.2.1 for a formal definition). Then, $\boldsymbol{\pi}_{\mathsf{ElG\text{-}com}}$ satisfies correctness, hiding (under the DDH assumption), and binding according to Definitions 5.4.1 to 5.4.3, respectively.*

Proof (Sketch). It is easy to see that the our correctness condition holds. Furthermore, with the simulator $\sigma^{\mathsf{B}}_{\mathsf{com}}$ outputting a random triple of group elements as commitment string, hiding holds under the DDH assumption, i.e., for ϵ encoding an appropriate reduction to the DDH problem. Finally, consider the function f that maps $(U, V, W) \in \mathbb{G}^3$ to $W \cdot g^{-\mathrm{DL}_g(U) \cdot \mathrm{DL}_g(V)}$ and all other bit-strings to $\perp$. For this function, it is easy to see that a simulator $\sigma^{\mathsf{A}}_{\mathsf{com}}$ exists such that the construction that

formalizes binding holds. See Appendix B.2.1 for a formal description of the respective simulators. □

5.4.2 Coin-Tossing

In this section, we consider Blum's simple coin-tossing protocol [Blu83]. The protocol assumes to have a commitment resource from Alice to Bob, and a communication channel in the reverse direction, at its disposal. It then proceeds as follows: Alice chooses $X \in \{0,1\}$ uniformly at random and commits to it. Once Bob is sure that Alice committed, he chooses $Y \in \{0,1\}$ uniformly at random and sends it over to Alice (in clear). Finally, Alice opens the commitment and both parties output $Z = X \oplus Y$.

Clearly, this protocol does not provide fairness—even when instantiated with a UC-secure commitment. This is due to the fact that both parties can always choose to abort the protocol by not responding, and in particular Alice can do so *after* she has seen the result. When instantiating the commitment with the resource constructed in the last section, one even obtains a weaker resource. Note that this is inherent for our construction being in the plain model, as otherwise it could be used as the bit of a CRS, contradicting well-known impossibility results.

In a nutshell, the resource obtained by our construction guarantees that the output is not biased, but does not exclude that during the opening phase, one of the parties learns some trapdoor allowing it to distinguish it from a uniformly random value. For example, our formalization would allow the resulting bit to be the first bit of a PRG's output, while leaking the seed during the opening phase. Note that such a coin toss resource is still useful, for instance for lotteries. First, if the resulting bit is just used to determine which party gets some good, then bias-resistance is obviously good enough irrespective of the fact that the parties might be aware that the result is only pseudo-random. Second, in a simple lottery where people's preferences are obvious, fairness can be achieved by declaring the party that caused the abort to have lost.

The coin-toss resource

The ideal specification is expressed in terms of the resources $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\mathcal{M}}$ and $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\mathcal{M},f}$, where the former denotes a restricted version of the latter. The

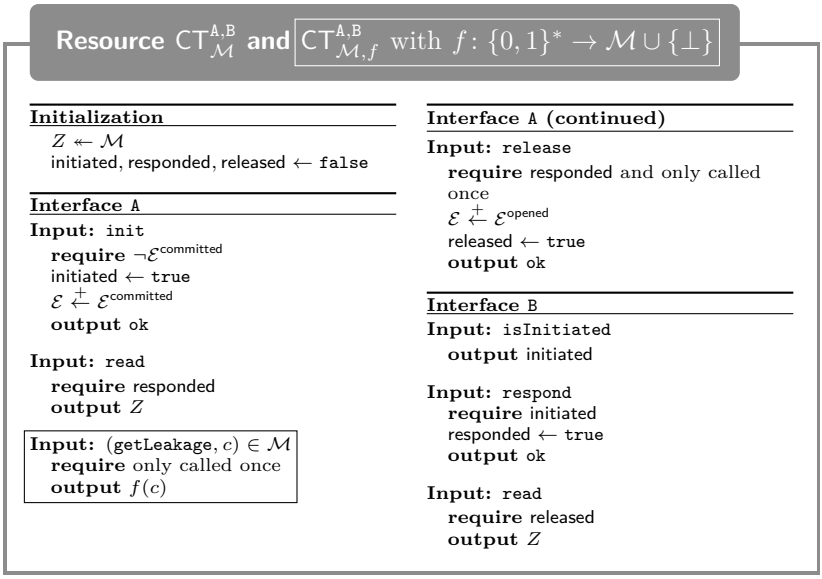


Figure 5.5: The coin-toss resources for coin space $\mathcal{M}$. In the unfiltered version, Alice additionally has the capability to once obtain a leakage to $f(c)$, where f is a parameter of the resource. Note that neither version provides fairness, as Alice can always chooses to not release the value after having seen it.

resource $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\mathcal{M}}$ initially draws an element $Z \in \mathcal{M}$ uniformly at random. In order for the coin-toss Z to become available to the parties, A has to initiate it, and B has to respond afterwards. From this point on, A can obtain Z and then decide whether the value should also be released to B. In the resource $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\mathcal{M},f}$, A furthermore can query once a leakage $f(c)$, of some potentially inefficient function f. A formal definition of the resources can be found in Figure 5.5.

The constructions

First, consider correctness. It is easy to see that the following construction holds, i.e., two honest parties actually get to agree on a uniform

random bit, as expressed in the following proposition.

Proposition 5.4.5. *Let* $\boldsymbol{\pi}_{\mathsf{CT}} \coloneqq (\pi^{\mathtt{A}}_{\mathsf{CT}}, \pi^{\mathtt{B}}_{\mathsf{CT}})$ *denote the pair of converters implementing Blum's protocol (cf. Appendix B.2.2 for a formal definition). Then, we have*

$$\left[\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\{0,1\}}, \mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}\right] \xmapsto{\boldsymbol{\pi}_{\mathsf{CT}}} \mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\}},$$

and thus

$$\left[\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_{1}, \mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_{2}, \mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}\right] \xmapsto{\boldsymbol{\pi}_{\mathsf{CT}} \circ \boldsymbol{\pi}_{\mathsf{com}}} \mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\}},$$

for any commitment scheme $\boldsymbol{\pi}_{\mathsf{com}}$ *satisfying Definition 5.4.1 (correctness).*

Second, consider the guarantee for an honest initiator $\mathtt{A}$.

Proposition 5.4.6. *Let* $\boldsymbol{\pi}_{\mathsf{CT}} \coloneqq (\pi^{\mathtt{A}}_{\mathsf{CT}}, \pi^{\mathtt{B}}_{\mathsf{CT}})$ *denote the pair of converters implementing Blum's protocol. Then, there exists an efficient simulator* $\sigma^{\mathtt{B}}_{\mathsf{CT}}$ *(connecting to Bob's interfaces only) such that*

$$\left[\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\{0,1\}}, \mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}\right] \xmapsto[\mathsf{sim}]{\pi^{\mathtt{A}}_{\mathsf{CT}}, \sigma^{\mathtt{B}}_{\mathsf{CT}}, 0} \mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\}},$$

and thus, for any commitment scheme $\boldsymbol{\pi}_{\mathsf{com}}$ *satisfying Definition 5.4.2 (hiding), we have*

$$\left[\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_{1}, \mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_{2}, \mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}\right] \xmapsto[\mathsf{interval}]{\pi^{\mathtt{A}}_{\mathsf{CT}} \circ \pi^{\mathtt{A}}_{\mathsf{com}}, \sigma^{\mathtt{B}}_{\mathsf{com}} \circ \sigma^{\mathtt{B}}_{\mathsf{CT}}, \tilde{\epsilon}, [\mathtt{true}, \mathcal{E}^{\mathsf{opened}}]} \mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\}},$$

with $\tilde{\epsilon} \coloneqq \left(\epsilon_{\sigma^{\mathtt{B}}_{\mathsf{CT}}}\right)_{[\mathtt{true}, \mathcal{E}^{\mathsf{opened}}]}$.

Proof. Recall that $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\{0,1\}}$ only reveals the value X to Bob after he sent his value Y. Hence, X and Y are independent and with X chosen uniform at random by Alice, implying that $Z = X \oplus Y$ is a uniform random value. Hence, using the simple simulator $\sigma^{\mathtt{B}}_{\mathsf{CT}}$ that simulates the output of the commitment resource as $X \coloneqq Z \oplus Y$ (see Appendix B.2.2 for a formal definition), it is easy to see that the construction actually achieves the coin-toss resource perfectly. □

Note that this implies that the output Z that Alice obtains looks indistinguishable from a uniform random value until the value is released

for the dishonest party. Hence, while it is not guaranteed that the dishonest party does not learn some trapdoor afterwards, the value Z is at least unbiased.

Finally, consider the security guarantees for an honest party $\mathtt{B}$ against a potentially dishonest party $\mathtt{A}$. To this end, we turn to the unfiltered resources $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\{0,1\},f}$ and $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\},f}$, where the latter once allows Alice to obtain $f(c)$ for a c of her choice.

Proposition 5.4.7. *Let $\boldsymbol{\pi}_{\mathsf{CT}} \coloneqq (\pi^{\mathtt{A}}_{\mathsf{CT}}, \pi^{\mathtt{B}}_{\mathsf{CT}})$ denote the pair of converters implementing Blum's protocol. Then, there exists an efficient simulator $\sigma^{\mathtt{A}}_{\mathsf{CT}}$ such that*

$$\left\{\left[\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}_{\{0,1\},f}, \mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}\right] \;\middle|\; f\colon \{0,1\}^* \to \{0,1,\bot\}\right\} \xrightarrow[\mathsf{sim}]{\pi^{\mathtt{B}}_{\mathsf{CT}}, \sigma^{\mathtt{A}}_{\mathsf{CT}}, 0} \left\{\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\},f} \;\middle|\; f\colon \{0,1\}^* \to \{0,1,\bot\}\right\},$$

and thus, for any commitment scheme $\boldsymbol{\pi}_{\mathsf{com}}$ satisfying Definition 5.4.3 (binding), we have

$$\left[\mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_1, \mathsf{Ch}^{\mathtt{A}\to\mathtt{B}}_2, \mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}\right] \xrightarrow[\mathsf{sim}]{\pi^{\mathtt{B}}_{\mathsf{CT}} \circ \pi^{\mathtt{B}}_{\mathsf{com}}, \sigma^{\mathtt{A}}_{\mathsf{com}} \circ \sigma^{\mathtt{A}}_{\mathsf{CT}}, 0} \left\{\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\},f} \;\middle|\; f\colon \{0,1\}^* \to \{0,1,\bot\}\right\}.$$

Proof. Consider the real-world system resulting from attaching Bob's converter only, for some function f. Interacting with this resource, the environment can input a commitment string C at Alice's interface, then see Bob's bit Y at Alice's channel interface, and finally see the resulting bit $Z = f(C) \oplus Y$ as the output of Bob's converter. In the following, consider the ideal-world system with the same function f as in the real world. It is now easy to see that a simulator can easily replicate the real-world behavior by getting Z from the resource, querying the leakage-oracle on C getting $f(C)$, and then setting $Y = Z \oplus f(C)$. A formal definition of the simulator $\sigma^{\mathtt{A}}_{\mathsf{com}}$ can be found in Appendix B.2.2. □

As a final note, observe that formalizing Bob's security guarantees for the commitment resource in terms of an interval-wise relaxation, rather than introducing the unfiltered resource $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\},f}$, would not work. This is due to the fact that in the real world Y (requiring the additional capabilities to simulate) is output at Alice's interface before

Bob sees Z. Hence, simulating only until Alice sends Y would not give any guarantees on Bob's output. In summary, this demonstrates Constructive Cryptography's advantage of being able to consider different types of statements within one (meta-)framework.

5.5 Revisiting Composable Identity-Based Encryption

In this section, we reexamine a result by Hofheinz, Matt, and Maurer [HMM15] implying that IND-ID-CPA security is not the right notion for identity-based encryption, unmasking this claim as an unnecessary framework artifact.

5.5.1 Background and Motivation

Identity-based encryption (IBE) is a generalization of public-key encryption that allows to encrypt messages using a master public key and the identity of the receiver, e.g., the e-mail address. This stands in contrast to a regular public-key encryption scheme, where the encryption needs the receiver's public key, suggesting IBE as a solution to the key-distribution problem.

An IBE scheme $\mathsf{IBE} \coloneqq (\mathsf{Gen}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ consists of four algorithms. The key generation algorithm $(mpk, msk) \leftarrow \mathsf{Gen}(1^\lambda)$ outputs a master public and a master secret key (given the security parameter as input). The extraction algorithm $sk_{\mathsf{id}} \leftarrow \mathsf{Ext}(msk, id)$ outputs a user secret key given the master secret key and the user's identity. Encryption $c \leftarrow \mathsf{Enc}(mpk, id, m)$ outputs a ciphertext, and $m' \leftarrow \mathsf{Dec}(sk_{id}, id, c)$ the corresponding plain-text. For correctness, it is required that for all $(mpk, msk) \leftarrow \mathsf{Gen}()$, all identities id, all messages m, and all $sk_{id} \leftarrow \mathsf{Ext}(msk, id)$, we always have $\mathsf{Dec}(sk_{id}, id, \mathsf{Enc}(mpk, id, m)) = m$. Security, on the other hand is classically formalized via game-based definitions. For security against passive attacks, the standard notion is ind-id-cpa, as depicted in Figure 5.6. Weaker notions have been proposed as well, such as a version ind-sid-cpa, depicted in Figure 5.6 as well, where the adversary has to choose the identity under attack without knowing the master public key.

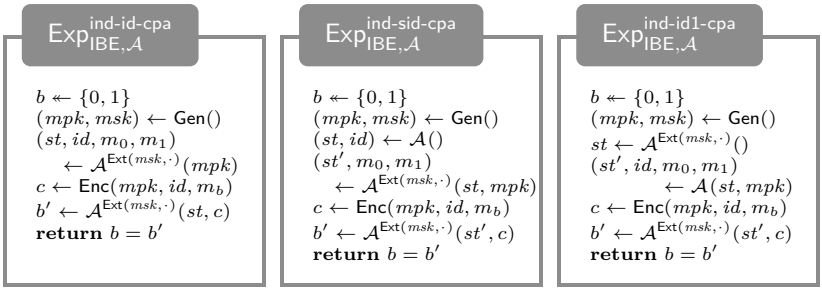


Figure 5.6: Game-based security definitions of identity-based encryption against passive attacks. The standard version (left) has been put forward by Boneh and Franklin [BF01]. Later, a weaker static notion (middle) has been introduced by Canetti, Halevi, and Katz [CHK03]. Finally, Hofheinz, Matt, and Maurer proposed a version loosely corresponding to the standard one under "lunchtime attack" (right). Note that in all of them, $\mathcal{A}$ is not allowed to query Ext on the identity it output.

Hofheinz, Matt, and Maurer [HMM15] investigated the composable security of IBE in Constructive Cryptography from an application centric point of view. To this end, they considered non-interactive communication, the apparent standard application of IBE. Assume that there is a trusted party that stores the master secret key and handles registration, i.e, hands out the user secret keys to the correct users. A set of users, each knowing the master public key and his user secret key(s), can now confidentially send messages to each other by encrypting it under the receiver's identity. The obvious security goal is that only the legitimate receiver can read the message. In turned out, however, that the standard ind-id-cpa security does not imply such a construction in the standard simulation-based construction notion—even when considering *static corruptions* only—due to the commitment problem. They, however, managed to show that such a scheme suffices in a weaker setting where it is guaranteed that all identities are registered before the first ciphertext is ever sent. Furthermore, the authors also introduced a new weaker security notion ind-id1-cpa, that essentially considers "lunchtime attack" and proved that it suffices for the same construction. They hence concluded, that the standard notion is at the same time both too strong (to achieve the weaker construction) and too

weak (to achieve the desired construction).

Since the weaker construction is not very realistic, e.g., in a company it is natural that new employees join long after the first ciphertext has been sent, the question about the right security definition remained open. In the remainder of this section, we devise a natural composable formalization based on interval-wise guarantees whose security exactly corresponds to the standard ind-id-cpa notion—resolving the issue.

5.5.2 The Real and Ideal Worlds

On a high level, Hofheinz et al. considered non-interactive secure communication in a setting with one honest sender $\mathtt{A}$, n potentially dishonest receivers $\mathtt{B}_i$, and one honest party $\mathtt{C}$ deriving and distributing the user secret keys. We here consider the same setting with essentially the same resources as in the original work.

In the real world, we assume that the sender $\mathtt{A}$ has a broadcast channel BCAST_n available, through which the ciphertexts are sent. For simplicity, we will assume guaranteed delivery throughout the rest of the example. Furthermore, we assume the existence of an authenticated channel $\mathsf{AUTH}_{\mathcal{P}}^{\mathtt{C}\to\mathtt{A}}$ from $\mathtt{C}$ to $\mathtt{A}$ to transmit the master public key Alice needs for encryption, and n secure channels $\mathsf{SEC}_{\mathcal{P}}^{\mathtt{C}\to\mathtt{B}_i}$ from $\mathtt{C}$ to $\mathtt{B}_i$ to transmit the user secret keys. A formal description of the corresponding resources can be found in Figure 5.7. Recall that we consider static corruptions here. Hence, the set of corrupted parties $\mathcal{P}$ appears as an explicit parameter of the resources.

The protocol securing the communication works as expected: whenever Alice wants to send a message to a certain identity, she encrypts it under the given *id* and broadcasts the ciphertext together with the identity. Each honest receiver then checks whether he has the corresponding decryption key, i.e., has been registered for this identity, and either decrypts the message or discards it. Finally, Charlie's protocol not only sends the master public key to Alice, but also allows to register identities for each receiver. Note that each identity can in principle be registered to many interfaces, i.e, many parties can posses the same user secret key. Furthermore, the assignment is not fixed but chosen by the environment, modeling an arbitrary or even adversarially chosen assignment. For completeness, a formal description of the corresponding converters $\pi_{\mathsf{Enc},t}^{\mathtt{A}}$, $\pi_{\mathsf{Dec}}^{\mathtt{B}_i}$, and $\pi_{\mathsf{Ext}}^{\mathtt{C}}$ is given in Figure 5.8.

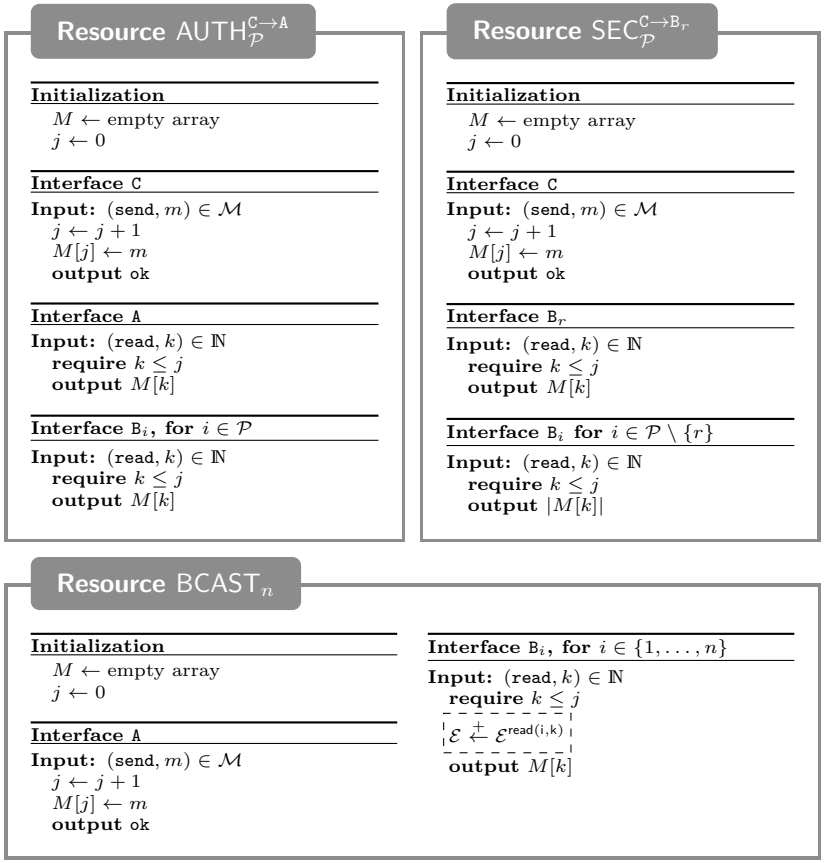


Figure 5.7: Description of the assumed resources.The set $\mathcal{P} \subseteq \{1, \ldots, n\}$ thereby specifies the set of statically corrupted receivers $\mathtt{B}_i$, while the sender A and registrar C are assumed to be honest.

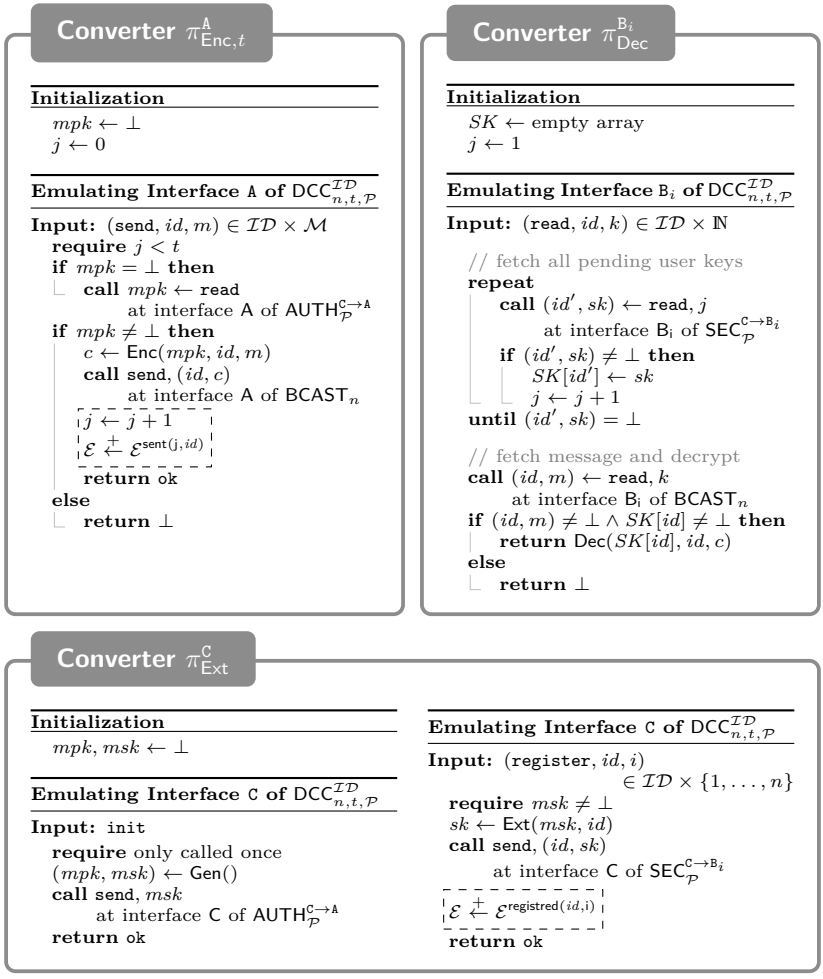


Figure 5.8: A formal description of the converters using IBE to achieve confidentiality.

Finally, consider the ideal resource $\mathsf{DCC}^{\mathcal{ID}}_{n,t,\mathcal{P}}$, which stands for *delivery controlled channel*. The name derives from the fact that the encryption can be seen as a mechanism with which the sender can specify to whom the message should be delivered. Hence, the resource acts like the broadcast channel, except that each message is only received by parties registered for the specified identity. Other dishonest parties may learn the message length, while honest parties ignore messages not intended for them. Furthermore, dishonest party might also learn which receiver is registered for identities (as the secure channel leaks the length of the user keys, which might depend on the identity). A formal definition of the resource is depicted in Figure 5.9. Note that compared to [HMM15], we will use global simulators rather than local ones, what allowed us to simplify the resource a bit.

5.5.3 The Composable Statement

One of the main results in [HMM15] has been showing that, due to the commitment problem, an IBE scheme does not construct the delivery controlled channel even under static corruptions.

Proposition 5.5.1 (Theorem 5.1 in [HMM15])**.** *Let $\mathcal{P} \subseteq \{1,\ldots,n\}$ denote the set of statically corrupted receivers. For every (efficient) simulator $\boldsymbol{\sigma}^{\mathcal{P}}_{\mathsf{IBE}}$, we have*

$$\left[\mathsf{BCAST}_n, \mathsf{AUTH}^{\mathtt{C}\to\mathtt{A}}_{\mathcal{P}}, \mathsf{SEC}^{\mathtt{C}\to\mathtt{B}_1}_{\mathcal{P}}, \ldots, \mathsf{SEC}^{\mathtt{C}\to\mathtt{B}_n}_{\mathcal{P}}\right] \xrightarrow[\mathsf{sim}]{\boldsymbol{\pi}^{n,t,\mathcal{P}}_{\mathsf{IBE}}, \boldsymbol{\sigma}^{\mathcal{P}}_{\mathsf{IBE}}, \epsilon_{\mathsf{ind\text{-}id\text{-}cpa}}} \mathsf{DCC}^{\mathcal{ID}}_{n,t,\mathcal{P}},$$

where $\epsilon_{\mathsf{ind\text{-}id\text{-}cpa}}$ denotes a reduction to the ind-id-cpa *game and*

$$\boldsymbol{\pi}^{n,t,\mathcal{P}}_{\mathsf{IBE}} := \left\{\pi^{\mathtt{A}}_{\mathsf{Enc},t}, \pi^{\mathtt{C}}_{\mathsf{Ext}}\right\} \cup \bigcup_{i\in\{1,\ldots,n\}\setminus\mathcal{P}} \pi^{\mathtt{B}_i}_{\mathsf{Dec}}$$

denotes the protocol where all honest parties apply their converter.

In the end, one is however not interested in such a strong guarantee. Rather, the above construction was just intended as a proxy to formalize confidentiality of the messages. Using our approach, hence immediately yields a better statement based on interval-wise guarantees.

Resource $\mathsf{DCC}^{\mathcal{ID}}_{n,t,\mathcal{P}}$

Initialization

$M \leftarrow$ empty array, $j \leftarrow 0$, initiated $\leftarrow$ `false`
for $i \in \{1, \ldots, n\}$ **do**
  $n_i \leftarrow 0$, $I_i \leftarrow$ empty array

Interface `A`

Input: $(\mathtt{send}, id, m) \in \mathcal{ID} \times \mathcal{M}$
require initiated $\wedge\ j < t$
$j \leftarrow j + 1$, $M[j] \leftarrow (id, m)$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{sent}(\mathtt{j}, id)}$
output `ok`

Interface `C`

Input: `init`
require only called once
initiated $\leftarrow$ `true`

Input: $(\mathtt{register}, id, i) \in \mathcal{ID} \times \{1, \ldots, n\}$
require initiated
$n_i \leftarrow n_i + 1$
$I_i[n_i] \leftarrow id$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{registred}(id, \mathtt{i})}$
output `ok`

Interface $\mathtt{B}_i$ **for** $i \in \{1, \ldots, n\}$

Input: $(\mathtt{read}, id, k) \in \mathcal{ID} \times \mathbb{N}$
require $k \leq j$
$(id, m) \leftarrow M[k]$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{read}(\mathtt{k}, \mathtt{i})}$
if $\exists \ell : I_i[\ell] = id \vee \mathcal{E}^{\mathsf{insec}(\mathtt{k})}$ **then**
  output (id, m)
else if $i \in \mathcal{P}$ **then**
  output $(id, |m|)$
else
  output $\perp$

Interface $\mathtt{B}_i$ **for** $i \in \mathcal{P}$ **(additional)**

Input: $(\mathtt{identities}, k) \in \{1, \ldots, n\}$
output I_k

Figure 5.9: The delivery controlled channel with an honest sender `A`, an honest registrar `C`, and n receivers $\mathtt{B}_i$, with $\mathcal{P} \subseteq \{1, \ldots, n\}$ denoting the statically corrupted ones.

More concretely, we consider each message individually and guarantee its confidentiality *until it trivially leaks* because a dishonest party is registered for the corresponding receiver's identity.

In the following, let $\mathcal{E}^{\mathsf{sent}(\mathrm{j},id)}$ indicate that the j-th message has been encrypted for identity id, $\mathcal{E}^{\mathsf{registred}(id,\mathrm{i})}$ indicate that the i-th receiver has been registered for identity id, and $\mathcal{E}^{\mathsf{read}(\mathrm{j},\mathrm{i})}$ that the i-th receiver read the j-th message. Moreover, let $\mathcal{E}^{\mathsf{insec}(\mathrm{j})}$ indicate that the j-th message is not confidential. See the dashed boxed in Figure 5.9 for a formal definition of when those events get triggered in the ideal world. Note that in the real world the former two events get triggered in the converters $\pi^{\mathtt{A}}_{\mathsf{Enc},t}$ and $\pi^{\mathtt{C}}_{\mathsf{Ext}}$, respectively (cf. Figure 5.8), whereas the third one gets triggered in the underlying broadcast channel BCAST_n (cf. Figure 5.7). This is due to the fact that Alice and Charlie are assumed to always apply their converter, while the dishonest receivers can access the broadcast channel directly.

Theorem 5.5.2. *Let $t \in \mathbb{N}$ denote an upper bound on the number of messages, let $\mathcal{P} \subseteq \{1,\dots,n\}$ denote an arbitrary set of statically corrupted receivers, and let $\boldsymbol{\pi}^{n,t,\mathcal{P}}_{\mathsf{IBE}}$ denote the protocol where all honest parties apply their converter as in Proposition 5.5.1. For each $\mathcal{P}$, there exists a sequence of efficient simulators $\boldsymbol{\sigma}^{\mathcal{P},j}_{\mathsf{IBE}}$ and a reduction $\epsilon_{\mathsf{ind\text{-}id\text{-}cpa}}$ to the* ind-id-cpa *game, such that*

$$\bigwedge_{j\in\{1,\dots,t\}} \Big(\big[\mathsf{BCAST}_n, \mathsf{AUTH}^{\mathtt{C}\to\mathtt{A}}_{\mathcal{P}}, \mathsf{SEC}^{\mathtt{C}\to\mathtt{B}_1}_{\mathcal{P}}, \dots, \mathsf{SEC}^{\mathtt{C}\to\mathtt{B}_n}_{\mathcal{P}}\big] \xrightarrow[\text{interval}]{\boldsymbol{\pi}^{n,t,\mathcal{P}}_{\mathsf{IBE}},\boldsymbol{\sigma}^{\mathcal{P},j}_{\mathsf{IBE}},\epsilon_{\mathsf{ind\text{-}id\text{-}cpa}},[P_{\mathsf{only}(\mathrm{j})},P_{\mathsf{leaked}(\mathrm{j})}]} \mathsf{DCC}^{\mathcal{ID}}_{n,t,\mathcal{P}} \Big),$$

where

$$P_{\mathsf{leaked}(\mathrm{j})}(\mathcal{E}) \coloneqq \exists i \in \mathcal{P} : \mathcal{E}^{\mathsf{sent}(\mathrm{j},id)} \wedge \mathcal{E}^{\mathsf{registred}(id,\mathrm{i})} \wedge \mathcal{E}^{\mathsf{read}(\mathrm{j},\mathrm{i})}$$

formalizes the event that the j-th message inherently leaked, and

$$P_{\mathsf{only}(\mathrm{j})}(\mathcal{E}) \coloneqq \bigwedge_{\ell\in\{1,\dots,t\}\setminus\{j\}} \mathcal{E}^{\mathsf{insec}(\ell)}$$

formalizes that we do not consider confidentiality of the other messages.

Converter $\sigma_{\mathsf{IBE}}^{\mathcal{P},j}$

Initialization

$(mpk, msk) \leftarrow \mathsf{Gen}()$

Emulating Int. $\mathtt{B}_i$ **of** BCAST_n, $i \in \mathcal{P}$

Input: $(\mathtt{read}, k) \in \mathbb{N}$
  if $M[k] = \bot$ **then**
    if $k \neq j$ **then**
      call $(id, m) \leftarrow \mathtt{read}, k$ at interface $\mathsf{B_i}$ of BCAST_n
      if $(id, m) \neq \bot$ **then**
        $M[k] \leftarrow (id, \mathsf{Enc}(mpk, id, m))$
    else
      call $(id, len) \leftarrow \mathtt{read}, k$ at interface $\mathsf{B_i}$ of BCAST_n
      if $(id, len) \neq \bot$ **then**
        $M[k] \leftarrow (id, \mathsf{Enc}(mpk, id, 0^{len}))$
  else
    $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{read(k,i)}}$
  return $M[k]$

Emulating Int. $\mathtt{B}_i$ **of** $\mathsf{AUTH}_{\mathcal{P}}^{\mathtt{C}\to\mathtt{A}}$, $i \in \mathcal{P}$

Input: $(\mathtt{read}, k) \in \mathbb{N}$
  if $k = 1 \wedge \exists id : \mathcal{E}^{\mathsf{sent}(1,id)}$ **then**
    return mpk
  else
    return $\bot$

Emulating Int. $\mathtt{B}_i$ **of** $\mathsf{SEC}_{\mathcal{P}}^{\mathtt{C}\to\mathtt{B}\ell}$, $i \in \mathcal{P}$

Input: $(\mathtt{read}, k) \in \mathbb{N}$
  call $I_\ell \leftarrow \mathtt{identities}, \ell$ at interface $\mathsf{B_i}$ of $\mathsf{DCC}_{n,t,\mathcal{P}}^{\mathcal{ID}}$
  if $I_\ell[k] = \bot$ **then**
    return $\bot$
  else
    $sk \leftarrow \mathsf{Ext}(msk, I_\ell[k])$
    if $\ell = i$ **then return** sk
    else return $|sk|$

Figure 5.10: The simulator from Theorem 5.5.2.

Proof (Sketch). The simulator initially generates a master secret-key and master public-key pair. Observe that for all messages except the j-th one, the ideal-world resource outputs the actual message (together with the corresponding identity). Hence, the simulator can simply encrypt it himself under the correct identity. For the j-th message, observe that the simulator only has to work if none of the dishonest receivers obtained the corresponding user key, and this key will then never get revealed. Hence, the simulator can encrypt the zero-string of the appropriate length. See Figure 5.10 for a formal definition of the simulators.

Observe that the reduction from distinguishing the real- and idea-world to the ind-id-cpa game is straight-forward. For all but the j-th message, the reduction can simply query the game for the corresponding user key, and the challenge can be chosen as $(m_j, 0^{|m_j|})$. If $b = 0$, this corresponds to the real-world behavior, and if $b = 1$ to the ideal-world behavior. □

Remark (On interpreting conjunction specifications). Observe that phrasing our statement for a bounded number of messages is without loss of generality in any actual application. We, however, have chosen to do so for good reasons. Assume for the moment that ϵ is simply a constant in $[0, 1]$. If we have a specification of the form $\bigcap_{i \in \mathcal{J}} \mathcal{S}_i{}^{\epsilon}$, then this can be read as the guarantee given by S_i holding with probability $1 - \epsilon$. In the end, we are however interested to know the probability such that all guarantees hold. In general, the best we can do is to apply the union bound, implying that the error is bounded by $\epsilon|\mathcal{J}|$. As a result, it is important that the number of conjunctions is small for the statement to be meaningful. Especially, while we could prove an analogous statement with intervals terminated by the leakage of each user key (rather than the messages), this would only make sense if the identity-space is small, in which case ind-sid-cpa security would suffice.

Chapter 6

A Case Study: Secure Messaging

In this chapter, we demonstrate how the tools and techniques developed in the previous parts apply to the context of secure messaging protocols. First, Constructive Cryptography with events allows us to deal with dependencies between various components and, thus, achieve proper decomposition of the protocols, which is a key requisite for modularization. Second, secure messaging protocols are prone to the simulator commitment problem, since they are typically designed for a setting where parties' state, including keying material, can be adaptively exposed.

6.1 Introduction

6.1.1 Motivation

Secure-messaging (SM) protocols attempt to provide strong security guarantees to two parties that communicate over an asynchronous network. Apart from protecting confidentiality and integrity of messages, the desired properties include forward secrecy and healing from a state or randomness exposure. The latter properties are addressed by the so-called ratcheting protocols, having the parties continuously update their secret keys. The term ratcheting on its own does not carry any formal

meaning; rather, it is an umbrella term for a number of different guarantees, somehow related to the concept of updating keys. One notable example of ratcheting is the widely-used Signal protocol [OWS17] with its double-ratchet algorithm, formally analyzed in [CCD+17; ACD19]. Furthermore, there exist protocols with much stronger guarantees, but that require the messages to be delivered in order [Poe18; JS18; DV18; JMM19a]. Protocols with the stronger guarantee of immediate out-of-order decryption have been proposed in [ACD19]. While the majority of the literature considers secure communication, some works view ratcheting as a property of key exchange instead [BSJ+17; Poe18].

A number of proposed protocols pursue similar goals, but each achieves a slightly different trade-off between security, efficiency and usability. Moreover, each construction comes with its own—usually fairly complex—security game, intermediate abstractions, and primitives. This renders them hard to compare and hinders achieving new trade-offs that would result from combining ideas from different protocols.

To remedy those issues, composable security frameworks appear to be a natural fit. Ideally, each sub-protocol would come with a composable security proof, forming a library of reusable statements that can be freely plugged together to achieve protocols that hit the right security-efficiency trade-offs for the given context. Traditionally, composable statements have been hindered, however, by two of the major obstacles considered in this thesis. First, secure messaging or ratcheting protocols are inherently designed for a setting where parties' state is assumed to be leakable, making them subject to the commitment problem. Second, due to their asynchronous nature, their security properties are typically intertwined, e.g., with post-compromise security heavily depending on the order of message delivery.

6.1.2 Contributions

Composable, unified, and modular statements. We present the first model of secure messaging (sub-)protocols in a composable framework. We thereby employ our solutions from Chapter 5 to overcome the commitment problem that is inherent to SM protocols designed for a setting where parties' state is assumed to be leakable. Furthermore, we use CC with events, introduced in Chapter 3, to disentangle dependencies among the various components in a clean yet systematic manner,

paving the way for a truly modular design and analysis.

Building on CC with events, we moreover parameterize resources by several discrete parameters that depend on the global event history. The goal of a (sub-)protocol is then understood as improving a certain parameter (e.g. making a channel confidential) while leaving the other parameters unchanged, independently of what they are, leading to statements that are reusable in different protocols or contexts. This approach stands in stark contrast to existing game-based definitions, which usually formalize exactly what is required by the next sub-protocol for the overall protocol's security proof to go through.

We consider three ratcheting sub-protocols. On the way, we discover cases where the existing game-based notions are insufficient to prove the stronger, more modular, statements that don't fix the properties (i.e., the switch positions) of the assumed network, but where they can be achieved by simple modifications.

A non-committing protocol. While our novel solutions from Chapter 5 allows us to circumvent the simulator commitment problem, sometimes one might still aim to achieve an even stronger specification. For instance, one might want to translate them to other simulation-based framework not allowing for such interval-wise specifications, or one just needs a stronger statement as a technical intermediate step. As a second contribution, we thus propose a technique that allows to transform many standard SM protocols into protocols that achieve full composable security, at the expense of an efficiency lost, as well as being restricted to only sending a bounded number of messages before receiving a reply from the other party. We apply this technique to the HIBE protocol mentioned above and construct its fully composable version.

6.1.3 The Constructive Cryptography Setting

In this chapter, we build on Constructive Cryptography with events, as introduced in Chapter 3. Since we consider secure messaging protocols for bilateral communication in the presence of a network adversary, we use the Alice-Bob-Eve setting, as introduced in Section 2.2.6, with $\mathcal{P} = \{\mathtt{A}, \mathtt{B}, \mathtt{E}, \mathtt{F}\}$ (where $\mathtt{F}$ denotes a free interface directly accessed by the distinguisher). An overview of this setting is depicted in Figure 6.1. We specify protocols (for the honest parties) as a tuple, i.e., $\boldsymbol{\pi} \coloneqq (\pi^{\mathtt{A}}, \pi^{\mathtt{B}})$,

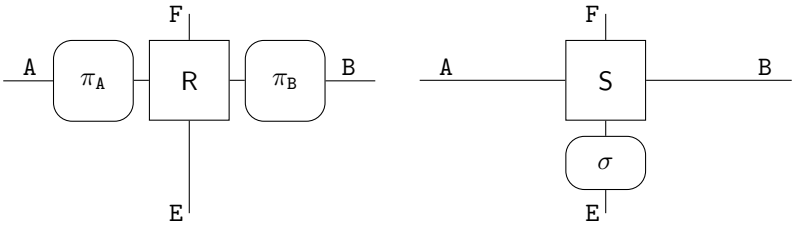


Figure 6.1: Execution of the protocol in the real world by Alice and Bob (left) and the ideal world with the simulator attached to Eve's interface (right). The free interface on the top is accessed directly by the environment in both worlds.

where the first converter is understood to be connected to Alice's interfaces and the second converter to Bob's interfaces, respectively.

All our security statements in this chapter involve computational security, i.e., ϵ-relaxations, and are simulation-based. We, thus, employ the construction notion introduced in Section 2.2.5.

6.2 The Unified Composable Statement

In this section, we introduce the unified type of construction statement that we make about SM protocols and components thereof.

6.2.1 The Approach

We opt for the natural choice of an *application-centric approach*, where the security of a cryptographic scheme or primitive is defined as the construction it achieves when used in a particular application. While this approach provides readily understandable and clean security statements, the resulting definitions often turn out to be overly specific. For instance, the statement about an encryption scheme might hard-code a particular assumed authentic communication network, implying that it cannot be directly combined with an authentication scheme achieving slightly different guarantees.

Avoiding such overly specific statements is crucial for a modular

treatment of ratcheting protocols, as each sub-protocol of the prior literature achieves slightly different guarantees. We address this problem by making parameterized construction statements, where the assumed real-world resources are parameterized by several "switches" determining their security guarantees. Formally, such a "switch" is represented by a function of the global event history $\mathcal{E}$ (among others), that dynamically defines the behavior of the resource at a given moment in time. For instance, a leakage function $\mathfrak{L}$ may specify to which extent a channel leaks depending on the set of events that happened so far. The goal of a protocol is then to improve certain parameters while leaving the others unchanged, independently of what they were in the beginning. That is, our construction statements will be of the type that a protocol constructs a communication network with certain (stronger) guarantees, assuming a network with certain (weaker) guarantees, where the real-world guarantees are treated as a parameter instead of hard-coding them.

Note that in the context of ratcheting protocols, making such parameterized statement about components—without a-priori assuming any guarantees about the real-world—is mostly not an issue. This is due to the fact that the protocols anyway have to be designed for the setting where the state and randomness could leak at any time, temporarily nullifying all guarantees that the component might try to assume from the underlying sub-protocols.

6.2.2 Our Channel Model

We now introduce our model of two-party communication networks. It allows us to express flexible security guarantees, but also various usability restrictions or guarantees, such as whether messages can be received out of order or not.

Many single-message channels. We choose to model the communication network between Alice and Bob as the parallel composition of many unidirectional single-message communication channels. Besides being simpler to describe, it allows to have simpler construction steps which only consider a subset of the channels. On the flip side, it results in a world with an arbitrary but bounded number of messages, as the set of resources is static in CC. This is, however, without loss of generality

as long as the protocols do not take advantage of this upper bound. Finally, observe that this decision results in a network where messages have implicit (unprotected) sequence numbers, as for instance achieved by TCP.

The single-message channel. We model channels with authenticated data. Since we will use the same type of channel both in the real and ideal world, the channel must hit the right trade-off between giving enough power to the simulator but not too much power to the real-world adversary.

On a high level, the channel interfaces and their capabilities are as follows. See Figure 6.2 for the formal definition.

- The sender `S` can issue the command $(\texttt{send}, m, ad)$. Whether she is allowed to do so is determined by the can-send predicate $\mathfrak{S}$. (This predicate will mainly be used to describe situations in which the sender does not have the necessary keys yet.) A successful sending operation triggers the event $\mathcal{E}^{\mathsf{sent}}$. The sender can also query whether the channel is available for transmission.

- The adversary `E` can then potentially learn m through the `read` command. Whether she is allowed to do so is determined by the can-leak function $\mathfrak{L}$, which outputs either `false` (the adversary is not allowed to read m), `true` (reading is allowed but triggers a leaked event $\mathcal{E}^{\mathsf{leaked}}$), or `silent` (reading is allowed). Moreover, she is always allowed to learn the length of m and the (non-confidential) associated data ad.

- The adversary decides when receiving becomes possible, i.e., the message in principle is delivered. Once this happens, the receiver `R` can try to fetch the message. This has two possible outcomes: either he receives a message and an according received event is triggered, or he receives $\perp$ and an error event (indexed by an error code from Errors) is triggered. Which case happens is determined by the delivery function $\mathfrak{D}$, which takes into account the event history and on whether the message that `R` tries to fetch is the same as the one input by `S` (or an injected value from the adversary). The latter condition is denoted by the flag *same*. The flag *same*

Resource $\mathsf{Ch}^{\mathsf{id},\mathtt{S}\to\mathtt{R}}_{\mathfrak{L},\mathfrak{I},\mathfrak{S},\mathfrak{R},\mathsf{Errors}}$

Parameters:

- Identity id (optionally), and interfaces $\mathtt{S}$ (sender) and $\mathtt{R}$ (receiver)
- Set of Errors that can occur
- Functions $\mathfrak{L}(\mathcal{E}) \in \{\mathtt{true}, \mathtt{false}, \mathtt{silent}\}$ (can leak), $\mathfrak{S}(\mathcal{E}) \in \{\mathtt{true}, \mathtt{false}\}$ (can send), $\mathfrak{R}(\mathcal{E}) \in \{\mathtt{true}, \mathtt{false}\}$ (can receive) and $\mathfrak{D}(\mathcal{E}, same) \in \mathsf{Errors} \cup \{\mathtt{msg}\}$ (delivery outcome)

Events: $\mathcal{E}^{\mathsf{sent}}$, $\mathcal{E}^{\mathsf{leaked}}$, $\mathcal{E}^{\mathsf{received}(same)}$, $\mathcal{E}^{\mathtt{error}(err,same)}$ for $same \in \{\mathtt{true}, \mathtt{false}\}$ and $err \in \mathsf{Errors}$

Initialization

$m_\mathtt{S}, ad_\mathtt{S}, m_\mathtt{R}, ad_\mathtt{R}, cmd, same \leftarrow \bot$

Interface S

Input: $(\mathtt{send}, m, ad) \in \mathcal{M} \times \mathcal{AD}$
require $cmd = \bot$
if $\neg\mathfrak{S}(\mathcal{E})$ **then output** $\bot$
$(m_\mathtt{S}, ad_\mathtt{S}) \leftarrow (m, ad)$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{sent}}$
output $\mathtt{ok}$

Input: $\mathtt{isAvailable}$
output $\mathfrak{S}(\mathcal{E})$

Interface R

Input: $\mathtt{receive}$
require only called once
if $\neg\mathfrak{R}(\mathcal{E}) \vee cmd = \bot$ **then**
  output $\bot$

// same messages (no injection)?
if $same = \mathtt{check}$ **then**
  $same \leftarrow ((m_\mathtt{R}, ad_\mathtt{R}) = (m_\mathtt{S}, ad_\mathtt{S}))$

// outcome: an error or the message
$out \leftarrow \mathfrak{D}(\mathcal{E}, same)$
if $cmd = \mathtt{dlv} \wedge out = \mathtt{msg}$ **then**
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{received}(same)}$
  output $(m_\mathtt{R}, ad_\mathtt{R})$
else if $cmd = (\mathtt{err}, err, Overw) \wedge (out = \mathtt{msg} \vee out \in Overw)$ **then**
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathtt{error}(err,same)}$
  output $\bot$
else
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathtt{error}(out,same)}$
  output $\bot$

Input: $\mathtt{isAvailable}$
output $\mathfrak{R}(\mathcal{E}) \wedge cmd \neq \bot$

Interface E

Input: $\mathtt{read}$
if $\mathfrak{L}(\mathcal{E}) = \mathtt{false}$ **then output** $\bot$
else if $\mathfrak{L}(\mathcal{E}) = \mathtt{true}$ **then** $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}$
output $(m_\mathtt{S}, ad_\mathtt{S})$

Input: $\mathtt{readLength}$
output $(|m_\mathtt{S}|, ad_\mathtt{S})$

Input: $(\mathtt{deliver}, m, ad, same') \in (\mathcal{M} \cup \{\mathtt{fwd}\}) \times \mathcal{AD} \times \{\mathtt{check}, \mathtt{false}\}$
require $cmd = \bot$

// handle forwarding request
if $m = \mathtt{fwd}$ **then**
  if $m_\mathtt{S} = \bot$ **then output** $\bot$
  else $m \leftarrow m_\mathtt{S}$

// store for receiving
$(m_\mathtt{R}, ad_\mathtt{R}, same) \leftarrow (m, ad, same')$
$cmd \leftarrow \mathtt{dlv}$
output $\mathtt{ok}$

Input: $(\mathtt{error}, err, Overw, m, ad, same') \in \mathsf{Errors} \times 2^{\mathsf{Errors}} \times \mathcal{M} \times \mathcal{AD} \times \{\mathtt{true}, \mathtt{false}, \mathtt{check}\}$
require $cmd = \bot$

// (m, ad) only to determine same
if $same' = \mathtt{check}$ **then**
  $(m_\mathtt{R}, ad_\mathtt{R}) \leftarrow (m, ad)$

// store for receiving
$same \leftarrow same'$
$cmd \leftarrow (\mathtt{err}, err, Overw)$
output $\mathtt{ok}$

Figure 6.2: The single-message channel.

is also exposed as part of the received or error event $\mathcal{E}^{\mathsf{received}(same)}$ or $\mathcal{E}^{\mathsf{error}(err,same)}$, respectively.

- When the adversary decides that receiving is possible, she has two options: schedule the delivery of (m', ad') (command `deliver`), or force an error $err \in \mathsf{Errors}$ to be triggered (command `error`). In the first case, she can also request to just forward the sender's message (if one exists), using $m' = \texttt{fwd}$. Moreover, for technical reasons[1], she can also insist that once the receiver fetches the message, $same = \texttt{false}$ is used even if the messages match. In case the adversary forces an error err and the outcome of receiving would anyway be a (different) error, the existing error can either be overwritten or preserved. She can control this by specifying a set $Overw$ of errors that should be overwritten.

A note on confidentiality. In our channel, the $\mathcal{E}^{\mathsf{received}(same)}$ and $\mathcal{E}^{\mathsf{error}(err,same)}$ events indicate whether the message that Eve injected was the same as the sender's. Since we assume that those events are in principle observable by everybody, including the adversary, those events can partially breach confidentiality if the communication is not properly authenticated. However, those events are crucial to phrase the post-impersonation guarantees of certain ratcheting protocols. In fact, in those protocols Eve could usually inject her own message (after exposing the sender's state), observe whether it causes the communication to break down, and thereby deducing whether the sender wanted to send the same message afterwards. Our events simply reflect this.

6.2.3 Memory and Randomness Resources

An integral part of secure messaging protocols is the assumption that the parties' state, and sometimes also randomness, can leak to the adversary. In Constructive Cryptography everything that can be accessible by multiple parties, here the honest party and Eve, must be modeled as a resource. As a consequence, all of our converters will be stateless

[1] The simulator might need this capability, e.g., if two (abstracted away) ciphertexts decrypt to the same message. Note that providing additional capabilities to the adversary in the real world only strengthens the statement and directly implies the construction where this capability is removed.

and deterministic. (Stateless means that the converter cannot keep state between two separate invocations at the emulated interfaces.) The statements will contain explicit memory and randomness resources instead. These resources are formally defined in Figure 6.3.

On a high level, we consider two types of memory resources: (1) an insecure memory $\mathsf{IMem}^{id,\mathtt{U}}$, and (2) a potentially secure memory $\mathsf{Mem}^{id,\mathtt{U}}$. The former is multiple-use and its current content is always available to the adversary. On the other hand, a secure memory can be written to at most once. It can also be securely erased at any later time. Moreover, it is parameterized by a can-leak predicate $\mathfrak{L}$, that specifies whether the content is available to the adversary. When the adversary successfully reads the contents, a leaked event is triggered.[2] Observe that each memory can leak independently, which leads to more fine-grained statements compared to prior work where it was usually assumed that either the entire state leaks or not (a state often consists of many secret keys from different sub-protocols, which we put in different memories). Nevertheless, it does not appear to incur additional significant complications.

Defining a potentially leakable randomness resource is a bit subtle. In principle, the idea is that the randomness can leak to the adversary *at the moment* it is used (modeling that it is sampled fresh at this point and is not stored) by the honest party. However, this cannot be directly expressed like this due to the activation model of the version of Constructive Cryptography used (recall that the output is given at the same interface the input was given). Hence, we model randomness resources that can be in one of two states: leakable or not (as specified by the flag *leaks*). If the can-leak predicate evaluates to true, the adversary can switch the state to leakable by sending `triggerLeaking`, which also triggers the leaked event. When the resource is used by the honest party, fresh randomness is sampled. Additionally, if at this time the state is leakable, then the sampled value is stored and the adversary can read it at any time afterwards.

[2]Rewritable secure memory can then be modeled as the parallel composition of many write-once memory cells. The memory requirement of a protocol is not determined by the number of such write-once memories, but rather by the maximal number of them in use at any time.

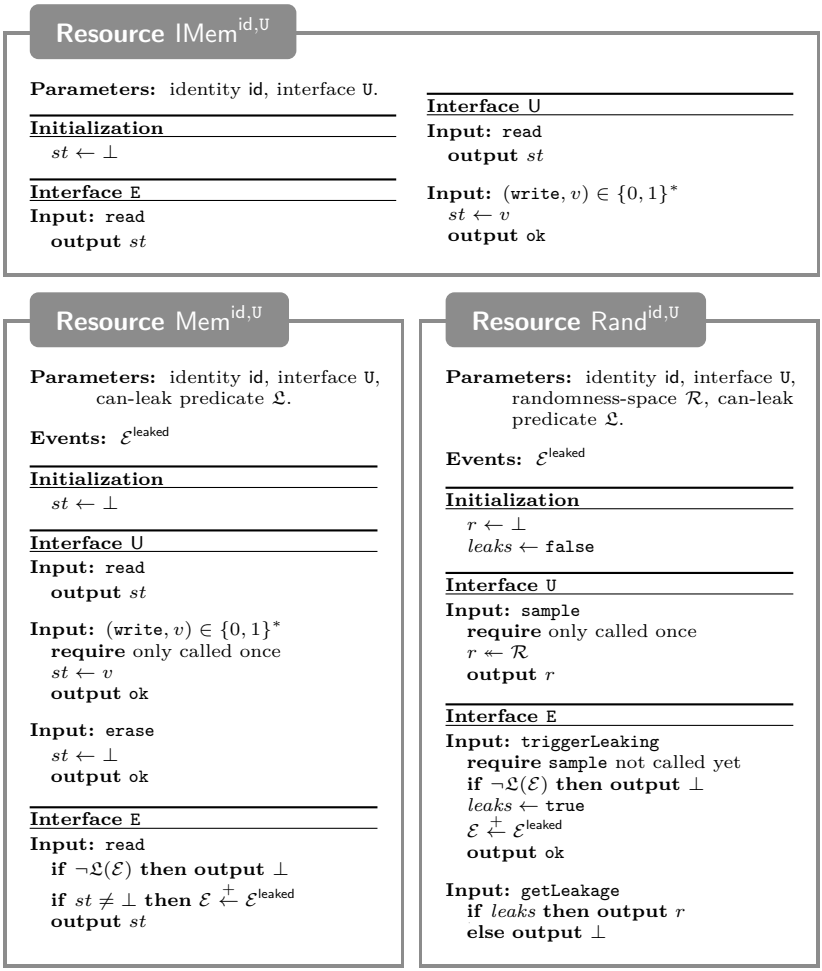


Figure 6.3: Formal definition of the memory and randomness resources.

6.3 Unifying Ratcheting: Two Examples

In this section, we get acquainted with how the security guarantees of ratcheting protocols can be phrased within our model. To this end, we model the guarantees of two components of actual ratcheting protocols.

As a first example, we consider a simple authentication scheme that appears in [JS18; DV18; JMM19a]. Using this example, we demonstrate how our framework allows for a fine-grained modularization, with the overall security then directly following from composition. As a second example, we consider the use of hierarchical identity-based encryption, as in [Poe18; JS18]. In this example, we moreover employ interval-wise guarantees to avoid the commitment problem of composable security.

6.3.1 A Simple Authentication Scheme

We first consider a simple unidirectional authentication protocol, which is designed with the strong guarantees of secure messaging in mind: the authentication guarantees should not only be forward secure but also heal after a state or randomness exposure of either party. Slight variations of this protocol have been used in [JMM19a] (without the hash) and [DV18] (using signcryption). Essentially the same idea also appeared in [JS18], where, however, a stronger signature primitive with updatable keys is considered, leading to the protocol being formalized in the bidirectional setting.

The protocol

In the protocol, whenever the sender wants to send a message, a fresh signing and verification key pair is sampled. The fresh verification key is then signed together with the message—using the prior signing key—and the message, the verification key and the signature are transmitted. Finally, the old signing key is securely erased and the fresh one stored instead. The receiver, on the other hand verifies a received message with the previous verification key and stores the new one. The scheme is depicted in Figure 6.4.

Recall that we aim to make a strong construction statement that considers how the scheme enhances any preexisting security guarantees, including confidentiality. Usually preserving confidentiality is not a goal

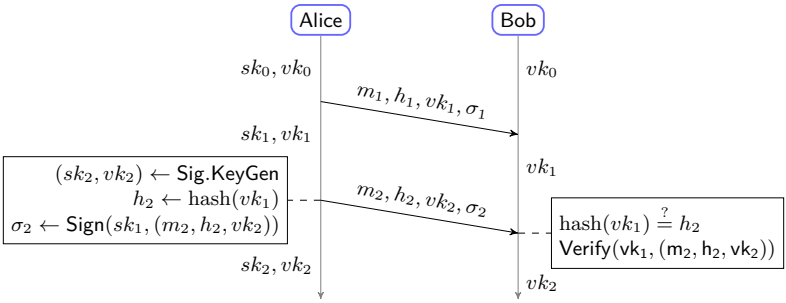


Figure 6.4: The simple scheme for unidirectional authentication.

that is considered for an authentication protocol, moreover, it is known that the authenticate-then-encrypt approach used in old versions of TLS is not generally secure [Kra01]. Nevertheless, we show that the scheme actually achieves this at the cost of assuming unique signatures instead of unforgeable ones (analogous to [JS18]), and with a minor modification: with each message, the sender also transmits a hash of the previous verification key. Such a hash is also present in the protocol from [JS18], and allows the receiver to check whether he is using the correct verification key.

The guarantees

Clearly, the protocol achieves authenticity if neither party's state is exposed. Moreover, Bob's state only consists of public information. If Alice's state gets exposed, then Eve obtains her current signing key that she can use to impersonate Alice towards Bob at this point in time. However, this key is useless to tamper with previous messages, even if they have not been delivered yet (forward security). More importantly, if, for some reason, Alice's next message containing a fresh verification key still is delivered without modification, then the signing key obtained by the adversary becomes useless thereby achieving the healing property. Hence, the adversary can inject the i-th message if and only if Alice's state between the $(i-1)$-st and i-th message got exposed, or there has already been a successful injection before.

Expressing the scheme's security guarantees in a game-based manner

turned out to be surprisingly involved compared to the scheme's simplicity and how easy it seems to intuitively describe its guarantees. Notably, to show its security, in [JMM19a] the abstraction of a key-updating signature scheme, as well as its corresponding correctness and security games, have been introduced. See Appendix C.1.1 for the corresponding definitions. This raises a couple of questions: can't we do simpler? What is the right security statement to make about this quite simple protocol, and what happens if the channel already provides certain authenticity or confidentiality guarantees? In the following, we try to answer these questions.

The construction statement

First, note that we consider the authentication of messages directly, and do not introduce an intermediate signature notion. Secondly, we consider authenticating the i-th message only, and to this end consider the $(i-1)$-st message where the fresh verification key is transmitted (we do not authenticate this message here) and the i-th message that is then signed under the corresponding signing key. Authenticating the $(i-1)$-st message, and all others, is then taken care of by iteratively applying the protocol, with the overall security directly implied by the composition theorem. This leads to the following real world resources

$$\mathsf{R}_i^{\mathsf{auth}} := \Big[\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}, \mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}, \mathsf{Rand}^{kg_i,\mathtt{A}}, \mathsf{Mem}^{sk_i,\mathtt{A}}, \mathsf{IMem}^{vk_i,\mathtt{B}}\Big], \quad (6.1)$$

where besides the two channels the sender also has a memory to store the new signing key, and the receiver a (insecure) memory to store the verification key. Furthermore, the sender also has an explicit randomness resource available (note that we only need key-generation randomness, since unique signatures are deterministic). The corresponding protocol $\mathbf{auth}_i := (\mathsf{sig}_i, \mathsf{vrf}_i)$, consisting of the pair of converters that are connected to Alice's and Bob's interfaces of $\mathsf{R}_i^{\mathsf{auth}}$, respectively, simply implement the previously described protocol. A formal description of those protocol converters can be found in Appendix C.1.2.

The goal of the protocol is then phrased as constructing the following ideal-world resource

$$\mathsf{S}_i^{\mathsf{auth}} := \Big[\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}, \mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}\Big], \quad (6.2)$$

in which the channels can also trigger an error sig-err, indicating that the signature verification failed, in addition to the errors from the real-world counterparts.

The authentication guarantees for the i-th channel can then be expressed via the following delivery-function, which guarantees that an injection attempt ($\neg same$) when the key is not known will causes a signature-verification error sig-err, and preserves preexisting authenticity (recall that $\tilde{\mathcal{E}} \coloneqq \tau(\mathcal{E})$ denotes the real-world's event history):

$$\mathfrak{D}^{\mathsf{S}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}, same) \coloneqq \begin{cases} err & \text{if } \mathfrak{D}^{\mathsf{R}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\tilde{\mathcal{E}}, same) = err \\ & \quad \wedge err \neq \mathtt{msg} \\ \mathtt{msg} & \text{else if } same \vee \mathcal{E}_i^{\mathsf{sk\text{-}known}} \\ \mathsf{sig\text{-}err} & \text{else} \end{cases} \tag{6.3}$$

where in a slight abuse of notation, we define a composed event $\mathcal{E}_i^{\mathsf{sk\text{-}known}}$, which is triggered as soon as it is not excluded that the signing key corresponding to Bob's verification key is known to Eve:

$$\mathcal{E}_i^{\mathsf{sk\text{-}known}} \coloneqq \mathcal{E}^{\mathsf{injected}}_{\mathsf{Ch}(i-1,\mathtt{A}\to\mathtt{B})} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(kg_i,\mathtt{A})} \vee \left(\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i-1,\mathtt{A}\to\mathtt{B})} \prec \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(sk_i,\mathtt{A})} \prec \mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}\right).$$

On the flip side, the scheme limits the availability of the channels to be sequential. While sending messages in order is natural for Alice, the protocol restricts Bob to receive them in order as well. We can express this using the following predicates.

$$\mathfrak{S}^{\mathsf{S}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}) \coloneqq \mathfrak{S}^{\mathsf{R}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\tilde{\mathcal{E}}) \ \wedge \ \mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i-1,\mathtt{A}\to\mathtt{B})}, \tag{6.4}$$

$$\mathfrak{R}^{\mathsf{S}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}) \coloneqq \mathfrak{R}^{\mathsf{R}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\tilde{\mathcal{E}}) \ \wedge \ \mathcal{E}^{\mathsf{received}}_{\mathsf{Ch}(i-1,\mathtt{A}\to\mathtt{B})}. \tag{6.5}$$

Note that our model simply forces us to make this restriction explicit, whereas this is often just hard-coded in games.[3]

All other predicates are preserved, e.g. $\mathfrak{L}^{\mathsf{S}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}) \coloneqq \mathfrak{L}^{\mathsf{R}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\tilde{\mathcal{E}})$. The security of the protocol can then be phrased as constructing the ideal world $\mathsf{S}_i^{\mathsf{auth}}$ from the real world $\mathsf{R}_i^{\mathsf{auth}}$, as summarized in the following theorem.

[3] Actually, many recently proposed secure-messaging protocols do have this restriction, which might limit their usability as pointed out by [ACD19].

Theorem 6.3.1. *Let* $\mathbf{auth}_i \coloneqq (\mathsf{sig}_i, \mathsf{vrf}_i)$ *be the authentication protocol described above. Let* $\mathsf{R}_i^{\mathsf{auth}}$ *be as in* (6.1)*, and let* $\mathsf{S}_i^{\mathsf{auth}}$ *be as in* (6.2)*, with the guarantees and restrictions as described in* (6.3)*,* (6.4)*, and* (6.5)*, respectively, and all others guarantees unchanged from* $\mathsf{R}_i^{\mathsf{auth}}$*. Then, if we map the event* $\mathcal{E}^{\mathsf{error}(\mathsf{sig\text{-}err}, same)}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$ *to* $\mathcal{E}^{\mathsf{received}(same)}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$*, we have*

$$\mathsf{R}_i^{\mathsf{auth}} \xmapsto[\mathsf{asympt}]{\mathbf{auth}_i} \mathsf{S}_i^{\mathsf{auth}},$$

if the underlying signature scheme is unforgeable with unique signatures, and the hash function is collision resistant.

Proof. The proof is found in Appendix C.1.3. Note that compared to a normal signature-scheme proof it is quite involved, which is the main price we pay for our much stronger statement.

Extending to many messages

So far, we only considered a world where Alice sends two messages, of which the second is authenticated. In a realistic setting, Alice can of course send many messages where all of them should be authenticated. In this section, we see how the composition theorem of Constructive Cryptography can be applied to directly get the desired result.

In particular, we start with a sequence of possibly unauthenticated channels $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$ for $i \in [n]$, where the authentication of $\mathsf{Ch}^{0,\mathtt{A}\to\mathtt{B}}$ can be seen as a setup assumption (it is standard to assume that Alice and Bob initially share a signing-verification key pair). Then, we iteratively apply the construction for two channels to $\mathsf{Ch}^{0,\mathtt{A}\to\mathtt{B}}$ and $\mathsf{Ch}^{1,\mathtt{A}\to\mathtt{B}}$, then to $\mathsf{Ch}^{1,\mathtt{A}\to\mathtt{B}}$ and $\mathsf{Ch}^{2,\mathtt{A}\to\mathtt{B}}$, etc. (cf. Figure 6.5). The composition theorem of CC guarantees that the composed protocol constructs the ideal world.

Corollary 6.3.2. *Let* $\mathsf{R}^{\mathsf{auth}}$ *and* $\mathsf{S}^{\mathsf{auth}}$ *denote the following real and ideal worlds*

$$\mathsf{R}^{\mathsf{auth}} \coloneqq \left[\left\{\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}\right\}_{i\in\{0,\ldots,n\}}, \left\{\mathsf{Mem}^{sk_i,\mathtt{A}}, \mathsf{IMem}^{vk_i,\mathtt{B}}\right\}_{i\in[n]}\right],$$

and

$$\mathsf{S}^{\mathsf{auth}} \coloneqq \left[\left\{\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}\right\}_{i\in\{0,\ldots,n\}}\right],$$

respectively. Moreover, let $\mathbf{auth} := ((\mathsf{sig}_1, \ldots, \mathsf{sig}_n), (\mathsf{vrf}_1, \ldots, \mathsf{vrf}_n))$ *denote the protocol attaching the* n *converters to Alice's and Bob's respective interfaces. Then, we have*

$$\mathsf{R}^{\mathsf{auth}} \xrightarrow[\mathsf{asympt}]{\mathbf{auth}} \mathsf{S}^{\mathsf{auth}},$$

where for each $i \in [n]$, $\mathfrak{I}^{\mathsf{S}^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$, $\mathfrak{S}^{\mathsf{S}^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$, *and* $\mathfrak{R}^{\mathsf{S}^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$ *are defined as in* (6.3), (6.4), *and* (6.5), *respectively.*

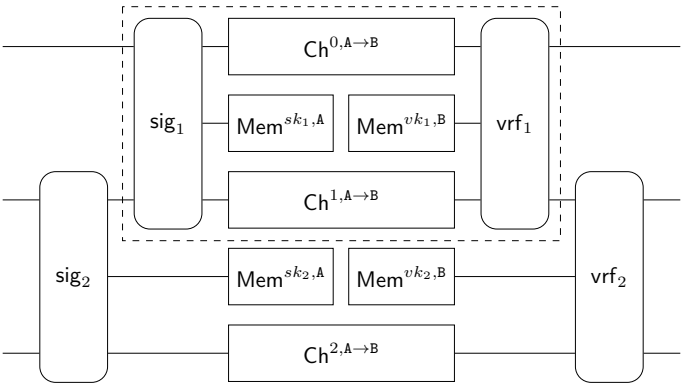


Figure 6.5: The first two steps constructing a sequence of authenticated channels: (1) The protocol $(\mathsf{sig}_1, \mathsf{vrf}_1)$ constructs a hybrid world, where the resources in the dashed box are replaced by two channels $\mathsf{Ch}^{0,\mathtt{A}\to\mathtt{B}}$ and $\mathsf{Ch}^{1,\mathtt{A}\to\mathtt{B}}$, where $\mathsf{Ch}^{1,\mathtt{A}\to\mathtt{B}}$ is authenticated as long as $\mathsf{Ch}^{0,\mathtt{A}\to\mathtt{B}}$ is. (2) $(\mathsf{sig}_2, \mathsf{vrf}_2)$ constructs the ideal world, where $\mathsf{Ch}^{1,\mathtt{A}\to\mathtt{B}}$ and $\mathsf{Ch}^{2,\mathtt{A}\to\mathtt{B}}$ are authenticated as long as $\mathsf{Ch}^{0,\mathtt{A}\to\mathtt{B}}$ is.

6.3.2 Confidentiality from HIBE

In the following we discuss a protocol from [JS18] that uses *hierarchical identity-based encryption (HIBE)* to add confidentiality to a sequence of channels. The protocol was designed for a challenging setting, where we do not assume authentication (as is usually done when talking about encryption). The reason is that in secure messaging authentication

cannot be guaranteed when the sender's state is exposed. This situation fits perfectly to our framework.

The protocol is described in the so-called sesqui-directional setting, introduced in [Poe18], meaning that the messages from both directions are considered, but only the guarantees of one of the directions are under concern—here from Alice to Bob. The bidirectional guarantees then follow directly from composition.

Hierarchical identity-based encryption

A HIBE scheme consists of the following four algorithms:

- A setup generation algorithm $(mpk, msk) \leftarrow \text{HIBE.Setup}(1^\kappa; r)$, generating the root master public and secret keys, i.e. $sk_{()} = msk$.
- A key-generation algorithm $sk_{\mathrm{id}\|id_n} \leftarrow \text{HIBE.Kgen}(sk_{\mathbf{id}}, id_n)$, where $(\mathbf{id} \parallel id_n) := (id_1, \ldots, id_{n-1}, id_n)$ for an identity vector $\mathbf{id} = (id_1, \ldots, id_{n-1})$.
- An encryption algorithm $c \leftarrow \text{HIBE.Enc}(mpk, \mathbf{id}, m; r)$.
- A decryption algorithm $m \leftarrow \text{HIBE.Dec}(sk_{\mathbf{id}}, c)$.

We require the HIBE scheme to be IND-CCA secure with certain additional properties that are not guaranteed by IND-CCA itself, but that most schemes do provide (see Appendix C.2.2 for details).

The protocol overview

On a high level, the protocol proceeds in epochs, where in each epoch Bob sends one message to Alice, and then Alice sends a sequence of messages to Bob. In particular, Bob's message contains a fresh HIBE public key mpk. For simplicity, consider the first epoch, as depicted in Figure 6.6. When Alice sends her i-th message, she encrypts it with mpk, using as the identity (the hashes of) all ciphertexts she sent before. Whenever Bob receives a ciphertext c_i, he decrypts it, derives the secret key for the new identity (with c_i appended) and erases the old key.

In the next epoch, Bob sends a new public key mpk', and we repeat. One subtle issue is how to run the epochs together. Note that, for example, Bob may send a number of public keys without receiving

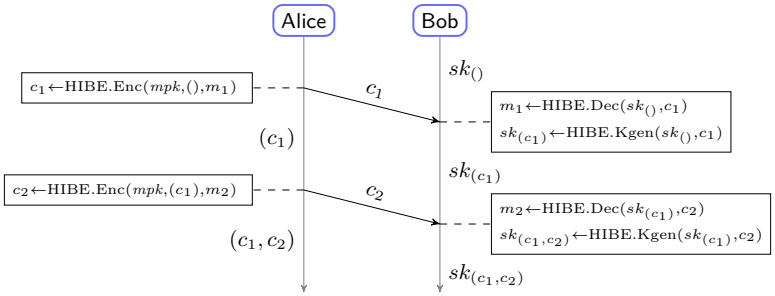


Figure 6.6: The first epoch of the sesquidirectional HIBE protocol.

a response, in which case he has to store secret keys from a number of epochs. A fresh secret key is stored for the empty identity, and when Bob receives a ciphertext, he updates all currently stored secret keys. This means that Alice uses for encryption of the i-th message a truncated transcript $(c_r, \ldots, c_{i-1})$. In order for her to compute it, Bob sends with each public key the index r of the last message he received. A graphical depiction of the full protocol can be found in Appendix C.2.1.

Security intuition

Intuitively, this use of HIBE allows to achieve three goals. The first is healing, achieved by exchanging fresh keys, as in most secure-messaging schemes. The second is forward secrecy: exposing the secret key after the i-th message is received does not affect the confidentiality of messages $m_1, \ldots, m_{i-1}$. This holds, since Bob updated all the secret keys with the identity c_i in the meantime. Healing and forward secrecy could also be achieved by a forward-secure PKE scheme. The last goal is the so-called post-impersonation security: an active injection destroys the decryption keys, so that its leakage exposes no messages. For this we need the hierarchy of identities. Roughly, injecting a message c'_i causes Bob to update his key to $sk_{(c_r,\ldots,c'_i)}$. This key gives no information about messages encrypted by Alice, since those will be for another identity $(c_r, \ldots, c_i)$.

The construction statement

To formalize these guarantees as a construction statement, we first have to describe the real world in which the protocol is executed. It consists of n channels from Alice to Bob (which the protocol protects) and n channels in the opposite direction on which the master public keys are transmitted. Moreover, Alice has memories to store the public keys and the transcript, and randomness resources for the encryption. Bob, on the other hand, has memories to store the secret keys and randomness resources for the key generation:

$$\mathsf{R}^{\mathsf{hibe}} := \Big[\big\{\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}\big\}_{i\in[n]}, \big\{\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}\big\}_{j\in[n]}, \mathsf{IMem}^{pk,\mathtt{A}}, \big\{\mathsf{Rand}^{kg_j,\mathtt{B}}\big\}_{j\in[n]}, \big\{\mathsf{Mem}^{tr_i,\mathtt{A}}, \mathsf{Rand}^{enc_i,\mathtt{A}}\big\}_{i\in[n]}, \big\{\mathsf{Mem}^{sk_{(j,i)},\mathtt{B}}\big\}_{j\in[n],i\in[n+1]\}}\Big], \quad (6.6)$$

where the index i indicates that the resource is related to transmitting the i-th message from Alice to Bob, and the index j indicates the j-th epoch. A formal description of the pair of converters implementing the protocol $\mathbf{hibe} := (\mathsf{hibe\text{-}enc}, \mathsf{hibe\text{-}dec})$ can be found in Appendix C.2.1.

The goal of the protocol is to enhance the confidentiality of the channels. Thus, the same set of channels is present in the ideal world, while the memory and randomness resources are used up:

$$\mathsf{S}^{\mathsf{hibe}} := \Big[\big\{\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}\big\}_{i\in[n]}, \big\{\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}\big\}_{j\in[n]}\Big]. \quad (6.7)$$

Moreover, the ideal channels can trigger an additional error $\mathsf{dec\text{-}err}$, indicating that decryption failed (this error event corresponds to the real-world delivery event when the adversary injects an invalid ciphertext).

We now proceed to formalize the confidentiality guarantees of $\mathsf{S}^{\mathsf{hibe}}$ by defining in which situations the i-th message might be known to the adversary:

The randomness leaked: If the encryption randomness leaked to the adversary, i.e., $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(enc_i,\mathtt{A})}$, then no PKE scheme can provide (full) confidentiality.

The master public key was set by Eve: If Alice encrypts using a master public key (potentially) set by Eve, Eve can trivially decrypt. That is, if Alice used the j-th master public key and $\mathcal{E}^{\mathsf{injected}}_{\mathsf{Ch}(j,\mathtt{B}\to\mathtt{A})}$.

The secret key leaked: Assume Alice sent the i-th message during the j-th epoch, and let $sk_{(j,i)}$ denote the secret key that Bob uses to decrypt that message. If Eve learned $sk_{(j,i)}$, the message is obviously not confidential, which either happens if the randomness used to generate the master secret key leaked or a key that allows to compute $sk_{(j,i)}$ leaked from Bob's memory:

$$\mathcal{E}^{\mathsf{sk\text{-}leaked}}_{i,j} := \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(kg_j,\mathtt{B})} \vee \exists k \in [r_j, i] : \Big(\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(sk_{(j,k)},\mathtt{B})} \wedge \forall \ell \in [r_j, k] : \neg\mathcal{E}^{\mathsf{injected}}_{\mathsf{Ch}(\ell,\mathtt{A}\to\mathtt{B})}\Big),$$

where r_j denotes the first message Bob received after sending the j-th public key (r_j is determined by the sent and received events in $\mathcal{E}$). Note that the last condition explicitly encodes the *post-impersonation guarantee*, meaning that $sk_{(j,k)}$ is only useful as long as Eve did not destroy it by injecting her own ciphertext. Forward-secrecy and healing, on the other hand, are encoded implicitly by the order in which those events can happen in the real world. We can make them more explicit by observing

$$\mathcal{E}^{\mathsf{sk\text{-}leaked}}_{i,j} \iff \mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(j,\mathtt{B}\to\mathtt{A})} \prec \mathcal{E}^{\mathsf{sk\text{-}leaked}}_{i,j} \prec \mathcal{E}^{\mathsf{received}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})},$$

where the former condition denotes healing and the latter forward-secrecy.

In summary, we can define the following event denoting that the i-th message is insecure

$$\mathcal{E}^{\mathsf{exposed}}_{i} := \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(enc_i,\mathtt{A})} \vee \mathcal{E}^{\mathsf{injected}}_{\mathsf{Ch}(j_i,\mathtt{B}\to\mathtt{A})} \vee \mathcal{E}^{\mathsf{sk\text{-}leaked}}_{i,j_i},$$

where j_i denotes the epoch in which the i-th message has been sent (which is computable from the order of events in $\mathcal{E}$), leading to

$$\mathfrak{L}^{\mathsf{S}^{\mathsf{hibe}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}) := \begin{cases} \mathtt{silent} & \text{if } \mathcal{E}^{\mathsf{exposed}}_{i} \\ \mathtt{false} & \text{otherwise.} \end{cases} \tag{6.8}$$

Notice that the above can-leak function fully overwrites any real-world guarantees, and silences the leaked events. This is because in the protocol Alice stores the communication transcript. As a consequence, when

her memory leaks, the ciphertext leaks as well, even if the assumed channel was in fact confidential. Moreover, this leakage does not correspond to the channel leaked event.

Analogous to the authentication scheme of the previous section, the HIBE scheme also limits the availability of the channels to be sequential, due to the hash-transcript used as identities. Moreover, Alice can obviously only encrypt using master public keys she received the public key. This could be made formal using the can-send and can-receive predicates $\mathfrak{S}$ and $\mathfrak{R}$, respectively, analogous to the previous section.

Avoiding the commitment problem

As described so far, the real and ideal world **hibe** $\mathsf{R}^{\mathsf{hibe}}$ and $\boldsymbol{\sigma}^{\mathrm{E}}\mathsf{S}^{\mathsf{hibe}}$, respectively, are easily distinguishable for any simulator $\boldsymbol{\sigma}^{\mathrm{E}}$ due to the commitment problem. Namely, if the distinguisher chooses to first see a ciphertext and then leak the corresponding decryption key, this cannot be simulated as the simulator first has to output a fake ciphertext, before getting to know the message, and then explain it by outputting a corresponding decryption key. For normal PKE, and especially HIBE, schemes this is impossible. Since we are nevertheless interested in composable expressing what the scheme actually does achieve, we apply our methodology of interval-wise guarantees from Chapter 5 to overcome the impossibility.

In principle on could provide very fine-grained guarantees using interval-wise guarantees: for each of the n individual messages, one could guarantee confidentiality until the adversary can trivially decrypt, using the intersection of n specifications. Here, we however opt fore a simpler construction statement that simply reflects the game-based definition. That is, we simply consider a single interval from the beginning of the experiment until any trivial impossibility causing the commitment problem. In summary, analogous to how the games disable certain oracles to prevent trivial impossibilities, we consider only guarantees until the adversary obtains the secret key To this end, we disallow the adversary from obtaining a secret key $sk_{(j,i)}$ that would allow to trivially identify a fake ciphertext. That is, we define the following indicator events

$$\mathcal{E}^{\mathsf{com\text{-}prob}}_{j,i} := \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(sk_{(j,i)},\mathsf{B})} \wedge \exists k > i : \left(\mathcal{E}^{\mathsf{committed}}_{k,j} \prec \mathcal{E}^{\mathsf{exposed}}_{k}\right), \tag{6.9}$$

where $\mathcal{E}^{\mathsf{committed}}_{i,j}$ denotes the event that the simulator commits on the i-th ciphertext, and that it was encrypted under mpk_j. More concretely, this happens if the distinguisher

- explicitly asked for the ciphertext;
- requested a hash-transcript that depends on the ciphertext;
- requested a secret key for which the identity depends on the ciphertext;
- actively injected a ciphertext that got decrypted under a secret key whose identity depends on the ciphertext under consideration,

leading to the following definition

$$\mathcal{E}^{\mathsf{committed}}_{i,j} \coloneqq (j_i = j) \wedge \Big(\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(tr_i,\mathtt{A})} \vee \Big(\neg\mathcal{E}^{\mathsf{injected}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \wedge \exists k \geq i : \big(\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(sk_{(j,k)})} \vee \mathcal{E}^{\mathsf{injected}}_{\mathsf{Ch}(k,\mathtt{A}\to\mathtt{B})}\big)\Big)\Big),$$

where again j_i denotes the epoch in which the i-th message has been sent.

Summary and analysis

The HIBE-based scheme achieves the so far described construction, with one exception: to provide more power to the simulator and make the construction statement provable, we need to silence the real-world channels' leaked events after the message is exposed, i.e, $\mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$ is arbitrary, except that if $\mathcal{E}^{\mathsf{exposed}}_i$, it no longer evaluates to `true`.[4]

Observe that while having to silence the leakage event in the real world limits reusability, the statement for instance is still generic enough to be composed with the authentication scheme from the previous section: if the real world is restricted like this (in the end, those events are just a mean to phrase dependencies and carry no real semantics), then the signature scheme, which preserves the can-leak predicate, and afterwards the HIBE scheme can be applied.

Overall, we have the following theorem, proved in Appendix C.2.2.

[4] This doesn't affect $\mathcal{E}^{\mathsf{committed}}_{i,j}$, that only considers leakage events before $\mathcal{E}^{\mathsf{exposed}}_i$.

Theorem 6.3.3. *Let* $\mathsf{R}^{\mathsf{hibe}}$ *be as in* (6.6) *with the restriction described above, and let* $\mathsf{S}^{\mathsf{hibe}}$ *be as in* (6.7) *with the confidentiality guarantees from* (6.8)*, and in-order sending and receiving. Moreover, let* $\mathcal{E}^{\mathsf{com\text{-}prob}} := \bigvee_{j,i} \mathcal{E}^{\mathsf{com\text{-}prob}}_{j,i}$ *denote the event that the commitment problem occurs, where* $\mathcal{E}^{\mathsf{com\text{-}prob}}_{j,i}$ *is as in Equation* (6.9)*. Then, if we map the event* $\mathcal{E}^{\mathsf{error}(\mathsf{dec\text{-}err},same)}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$ *to* $\mathcal{E}^{\mathsf{received}(same)}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$*, we have*

$$\mathsf{R}^{\mathsf{hibe}} \xrightarrow[\mathsf{interval\text{-}asym}]{\mathbf{hibe},[\mathtt{true},\mathcal{E}^{\mathsf{com\text{-}prob}}]} \mathsf{S}^{\mathsf{hibe}},$$

if the HIBE scheme is IND-CCA secure with our additional assumptions.

6.4 Adaptive Security

All protocols considered so far, and most of the ones in the literature, only achieve an interval-wise construction statement due to the commitment problem. While such a statement intuitively appears to be sufficient to guarantee confidentiality as well as authenticity of the messages, there might nevertheless be reasons to aim for the full traditional simulation-based security notion. For instance, (full) simulation-based security appears to be closely linked to deniability—another important property of secure messaging schemes.

Hence, in this section, we consider SM schemes that tolerate a fully adaptive adversary, i.e, allow to "explain" ciphertexts whenever needed due to leakage of secret keys. In particular, we present a technique that, given an SM protocol that is subject to the commitment problem, allows to construct an adaptive SM (ASM) protocol with almost the same guarantees, but that achieves fully adaptive security. This comes at the cost of efficiency and being able to send only a fixed number of messages without interaction. Applied to protocols with optimal security [JS18; Poe18], our technique enables even stronger guarantees.[5] As an example, we apply it to the HIBE protocol from Section 6.3.2.

Note that while the technique we use is essentially a general compiler that "removes" the commitment issue, formally phrasing such a theorem

[5] In game-based definitions, one can think of the "corrupt" oracle not being silenced even if the challenge has been issued, but instead outputting the secret state corresponding to the challenge bit 0.

would be rather cumbersome for at least two reasons. First, there is not just one game-based defintion of an SM scheme that could be lifted and, second, we require the specific simulation technique encoded in most game-based definitions, in contrast to the existential simulator of our constructive SM statements.

6.4.1 Overview

Receiver non-committing encryption. The technical tool we use to construct adaptively-secure secure-messaging (ASM) schemes with optimal security is so-called receiver non-committing encryption (RNCE), introduced by Canetti et al. [CHK05]. Intuitively, in RNCE schemes, key generation outputs an additional trapdoor z, ignored by honest parties and used by the simulator. Then, there are two ways to generate a ciphertext: (1) an "honest" ciphertext is computed in the standard way $c \leftarrow \mathsf{RNCE.E}(pk, m)$ (so, as in any encryption scheme, it is a commitment to the message), (2) a "fake" ciphertext is computed (by the simulator) without the message, but with the secret key sk and the trapdoor z as $\tilde{c} \leftarrow \mathsf{RNCE.F}(pk, sk, z)$. Given a fake ciphertext $\tilde{c}$ and any message m, one can compute a secret key $\tilde{sk} \leftarrow \mathsf{RNCE.R}(pk, sk, z, \tilde{c}, m)$ that explains the message-ciphertext pair (such that $\mathsf{RNCE.D}(\tilde{sk}, \tilde{c}) = m$). Moreover, the distributions (c, sk) (as in the real world) and $(\tilde{c}, \tilde{sk})$ (as in the simulation) are indistinguishable. This allows to explain a single ciphertext per public key.

The scheme. At a high level, the authors of [CHK05] use RNCE to construct non-committing forward-secure public-key encryption by encrypting with a standard forward-secure public-key scheme RNCE ciphertexts instead of messages. We generalize this idea (and the simulation technique) to SM protocols. In particular, we can construct an ASM scheme by taking a standard SM scheme that is subject to the commitment problem, i.e., satisfies only an interval-wise guarantee, and sending, instead of messages, their RNCE encryptions, where each message is encrypted with a different public key. When a message is received, the secret key is immediately deleted. (For the moment, assume that whenever Alice sends a message, an RNCE key pair is "magically" generated — Alice uses the public key, and the secret key immediately appears stored in Bob's state.) This way, the modified

scheme inherits all guarantees of the original SM scheme. Furthermore, it can be simulated in the adaptive setting, as we will see below.

Let us now address the problem of how the RNCE keys are distributed. One trivial solution would be to include ℓ key pairs as part of the setup: the parties send their ℓ public keys at the beginning over an authenticated channel. First, this way we can send only ℓ messages overall. But even worse, the RNCE keys do not heal: when the receiver is corrupted for the first time, the simulator can explain all messages sent so far, but it also has to commit to all RNCE secret keys. Hence, adaptive security is never restored. To deal with this, we use the technique used in all SM schemes: we send with each message an update, consisting of ℓ fresh RNCE public keys. In particular, Alice (Bob will proceed analogously) stores some public keys previously received from Bob. When she sends the i-th message, she RNCE-encrypts it with one of the unused public keys, generates ℓ new key pairs, stores the secret keys, and sends the RNCE ciphertext, the ℓ public keys and i to Bob over the channel constructed by an SM scheme. Bob stores the greatest index i he has seen so far. Whenever he sees a messsage with a greater i, he ignores all RNCE public keys he has and replaces them by the ℓ newly received ones. Unlike in the first trivial solution, in the above protocol adaptive security is restored as fast as possible: with the first new message delivered from the other party.

Simulation. We give an intuition of how the above protocol can be simulated. Assume that the SM scheme has the standard simulator, as hard-coded in most game-based definitions. In particular, he executes the protocol, and when a memory is exposed, he shows to the distinguisher the real state. For ciphertexts corresponding to confidential messages it shows encryptions of 0's, while for non-confidential ones it shows encryptions of the actual message.

In the adaptive setting, the real and the ideal world are easily distinguishable for that simulator. This is because when a message is sent as confidential, and later the memory is exposed, the distinguisher sees in the ideal world the encryption of 0's. However, we can fix this with our new scheme: the new simulator encrypts, instead of 0's, a fake RNCE ciphertext to generate a ciphertext corresponding to a confidential message. When a memory is corrupted, he receives the message (which, of course, can no longer be confidential) and computes

the fake RNCE secret key according to the fake ciphertext. RNCE guarantees that this is indistinguishable from the real world, where we have honest ciphertext and an honest key.

A note on efficiency. First, observe that using a symmetric non-committing encryption scheme, such as the one-time pad, instead of RNCE would not work. This is because in many SM schemes corrupting the sender has no effect on confidentiality, implying that upon such a corruption, the simulator needs to output a key of the symmetric non-committing scheme without knowing the messages (which trivially breaks against a distinguisher knowing the message).

Moreover, while our construction of using nested encryption appears to be redundant, it can be observed that using RNCE only would not suffice. This is because SM schemes can provide certain advanced *confidentiality* guarantees not achieved by RNCE alone. For example, the optimal schemes such as [JS18; Poe18] provide so-called post-impersonation guarantees: once the adversary injects a message to Bob (after corrupting Alice) and then corrupts Bob, all messages sent by Alice afterwards are confidential.

Limitations. Our protocol requires a fixed upper bound on the number of messages a party can send without interaction (in particular, after ℓ messages it needs a new set of public keys from the partner). Unfortunately, overcoming this seems unlikely with our approach. This is due to the impossibility result by Nielsen [Nie02b]. It essentially says that a non-committing non-interactive public-key encryption scheme requires that the length of a secret key is at least the overall length of all messages encrypted. This means that we would need non-committing encryption, where the public and secret keys are updated, in other words, a non-committing equivalent of HIBE. To the best of our knowledge, this does not exist yet.[6]

[6]Note that the impossibility of [Nie02b] also rules out a solution where Alice RNCE-encrypts for Bob a new RNCE secret key, used for the next message — this secret key would leave no space for the message.

6.4.2 Combining RNCE with HIBE

Recall that the HIBE protocol from Section 6.3.2 is designed for the sesqui-directional setting, where it protects the confidentiality of messages sent by Alice. In the protocol, Bob sends to Alice HIBE master public keys, which results in epochs. In epoch j, Alice uses the j-th master public key to encrypt her messages with the transcript as identity. In this section we consider the analogous setting for the ASM protocol, consisting of RNCE composed with HIBE. That is, Bob sends ℓ RNCE keys alongside the HIBE keys, and Alice uses them to additionally encrypt her messages.

Hence, for the ASM construction we need in the real world the additional randomness $\mathsf{Rand}^{renc_i,\mathtt{A}}$ for RNCE-encrypting the i-th message and $\mathsf{Rand}^{rkg_j,\mathtt{B}}$ for generating the j-th set of ℓ keys, compared to the real world from the HIBE protocol. Moreover, we have memories $\mathsf{Mem}^{rsk_{(j,k)},\mathtt{B}}$ for storing the k-th RNCE secret key, generated in epoch j, and insecure (rewritable) memories $\mathsf{IMem}^{rpk,\mathtt{A}}$ for storing the set of RNCE public keys. Overall, the real-world resources are as follows.

$$\mathsf{R}^{\mathsf{ad\text{-}hibe}} := \Big[\mathsf{R}^{\mathsf{hibe}}, \big\{\mathsf{Rand}^{renc_i,\mathtt{A}}\big\}_{i\in[n]}, \big\{\mathsf{Rand}^{rkg_j,\mathtt{B}}\big\}_{j\in[n]}, \mathsf{IMem}^{rpk,\mathtt{A}}, \big\{\mathsf{Mem}^{rsk_{(j,k)},\mathtt{B}}\big\}_{j\in[n],k\in[\ell]}\Big], \quad (6.10)$$

where $\mathsf{R}^{\mathsf{hibe}}$ should be understood as the same set of resources as in Section 6.3.2. The restrictions on those set of resources are dropped, on the other hand, since we no longer need work around the commitment problem. This implies, however, that we have to directly consider security of the overall compiled protocol, instead of using the construction statement for HIBE and composition.[7] A formal description of the protocol $\mathbf{rnce} := (\mathsf{rnce\text{-}enc}, \mathsf{rnce\text{-}dec})$ implementing the RNCE protocol on top of the HIBE protocol is given in Figure 6.7.

In the ideal world, we have the same $2n$ channels: $\mathsf{S}^{\mathsf{ad\text{-}hibe}} := \mathsf{S}^{\mathsf{hibe}}$. Most properties of the constructed channels are the same as in the HIBE construction. In fact, our adaptive protocol only affects (1) availability — only ℓ messages can be sent without interaction, and (2)

[7] In general, the simulator for the SM scheme simply does not output the secret state from the commitment-causing memories, and our ASM simulator cannot generate it himself, since this would be inconsistent with the rest of the SM simulation.

Converter rnce-enc

Emulating Int. $\mathtt{A}$ **of** $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}, i \in [n]$

Input: $(\mathtt{send}, m, ad) \in \mathcal{M} \times \mathcal{AD}$
  require only called once & isAvailable

  // Read and update the state
  call $(j, k, PK) \leftarrow \mathtt{read}$
    at interface $\mathtt{A}$ of $\mathsf{IMem}^{rpk,\mathtt{A}}$
  call $(\mathtt{write}, (j, k+1, PK))$
    at interface $\mathtt{A}$ of $\mathsf{IMem}^{rpk,\mathtt{A}}$

  // Encrypt end send
  call $r \leftarrow \mathtt{sample}$
    at interface A of $\mathsf{Rand}^{renc_i,\mathtt{A}}$
  $c \leftarrow \mathsf{RNCE.E}(PK[k], m; r)$
  call $s \leftarrow (\mathtt{send}, c, (ad, j, k))$
    at interface $\mathtt{A}$ of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  return s

Input: $\mathtt{isAvailable}$
  call $av \leftarrow \mathtt{isAvailable}$
    at interface $\mathtt{A}$ of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  call $(j, k, PK) \leftarrow \mathtt{read}$
    at interface $\mathtt{A}$ of $\mathsf{IMem}^{rpk,\mathtt{A}}$
  return $av \wedge (j, k, PK) \neq \bot \wedge k \leq \ell$

Emulating Int. $\mathtt{A}$ **of** $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}, j \in [n]$

Input: $\mathtt{read}$
  require only called once
  call $(m, (ad, PK)) \leftarrow \mathtt{read}$
    at interface $\mathtt{A}$ of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  call $(j', \_, \_) \leftarrow \mathtt{read}$
    at interface $\mathtt{A}$ of $\mathsf{IMem}^{rpk,\mathtt{A}}$

  // Update the state if the keys are newer than the stored ones.
  if $j = 1 \vee j' < j$ **then**
    call $(\mathtt{write}, (j, 1, PK))$
      at interface $\mathtt{A}$ of $\mathsf{IMem}^{rpk,\mathtt{A}}$
  return (m, ad)

Input: $\mathtt{isAvailable}$
  call $av \leftarrow \mathtt{isAvailable}$
    at interface $\mathtt{A}$ of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  return av

Converter rnce-dec

Emulating Int. $\mathtt{B}$ **of** $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}, j \in [n]$

Input: $(\mathtt{send}, m, ad) \in \mathcal{M} \times \mathcal{AD}$
  require only called once & isAvailable

  // Generate ℓ new key pairs
  call $r_1, \ldots, r_\ell \leftarrow \mathtt{sample}$
    at interface A of $\mathsf{Rand}^{rkg_j,\mathtt{B}}$
  for $k \in [\ell]$ **do**
    $(pk_k, sk_k) \leftarrow \mathsf{RNCE.G}(r_k)$
    call $(\mathtt{write}, sk_k)$
      at interface B of $\mathsf{Mem}^{rsk_{(j,k)},\mathtt{B}}$
  call $s \leftarrow (\mathtt{write}, m, (ad, [pk_1, \ldots, pk_\ell]))$
    at interface $\mathtt{A}$ of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  return s

Input: $\mathtt{isAvailable}$
  call $av \leftarrow \mathtt{isAvailable}$
    at interface $\mathtt{B}$ of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  return av

Emulating Int. $\mathtt{B}$ **of** $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}, i \in [n]$

Input: $\mathtt{read}$
  call $(c, (ad, j, k)) \leftarrow \mathtt{read}$
    at interface $\mathtt{A}$ of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  if $(c, (ad, j, k)) = \bot$ **then return** $\bot$

  // Get the secret key and decrypt
  call $sk \leftarrow \mathtt{read}$
    at interface B of $\mathsf{Mem}^{rsk_{(j,k)},\mathtt{B}}$
  if $sk \neq \bot$ **then**
    $m \leftarrow \mathsf{RNCE.D}(sk, c)$
    call $\mathtt{erase}$
      at interface $\mathtt{B}$ of $\mathsf{Mem}^{rsk_{(j,k)},\mathtt{B}}$
  if $m \neq \bot$ **then return** (m, ad)
  else return $\bot$

Input: $\mathtt{isAvailable}$
  call $av \leftarrow \mathtt{isAvailable}$
    at interface $\mathtt{B}$ of $\mathsf{Ch}^{i,\mathtt{B}\to\mathtt{A}}$
  return av

Figure 6.7: The RNCE part of the adaptively-secure protocol in the sesqui-directional setting.

confidentiality — we need to account for the additional randomness and memory resources. Recall that the epoch j_i in which message i is sent by Alice is determined by the sent and received events. With this, the restriction (1) can be expressed with the can-send and can-receive predicate in a straightforward way.

Let us now focus on confidentiality. Recall that in the HIBE protocol, the can-leak predicate was defined using the event $\mathcal{E}_i^{\mathsf{exposed}}$, denoting that the i-th message sent by Alice is inherently insecure. We modify this event to account for the additional resources used by RNCE. Specifically, the message is exposed if the RNCE-encryption randomness leaks: $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(renc_i,\mathtt{A})}$, or if the RNCE secret key leaks. The latter happens if Bob's key-generation randomness leaks: $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(rkg_{j_i},\mathtt{B})}$, or if the secret key memory leaks: $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(rsk_{(j_i,k_i)},\mathtt{B})}$, where the i-th message was the k_i-th one sent in its epoch. Overall, this leads to the following composed event:

$$\mathcal{E}_i^{\mathsf{exposed\text{-}ad}} := \mathcal{E}_i^{\mathsf{exposed}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(renc_i,\mathtt{A})} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rnd}(rkg_{j_i},\mathtt{B})} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(rsk_{(j_i,k_i)},\mathtt{B})}.$$

The leakage function $\mathfrak{L}^{\mathsf{S}^{\mathsf{ad\text{-}hibe}}}_{\mathsf{Ch}(\mathtt{A}\to\mathtt{B})}$ is then defined analogously to that of the HIBE construction `silent` in case of $\mathcal{E}_i^{\mathsf{exposed\text{-}ad}}$, and `false` otherwise. We stress that the need to include these additional cases only arises from our fine-grained modeling of memory and randomness. In reality, it makes sense to consider only one memory storing the whole secret state, only one randomness for RNCE and HIBE encryption, and so on. In such a model, the confidentiality of our adaptively secure scheme and the non-adaptive one would coincide.

The security of our composed protocol is summarized in the following theorem.

Theorem 6.4.1. *Let* $\mathsf{R}^{\mathsf{ad\text{-}hibe}}$ *be as in* (6.10)*, and let* $\mathsf{S}^{\mathsf{ad\text{-}hibe}}$ *be as in above with the described confidentiality guarantees, in-order sending and receiving, and the restriction to* ℓ *messages per epoch. Moreover, let* **rnce** $\circ$ **hibe** *denote the protocol that for each honest party first applies the respective* **hibe** *converter, and then the respective* **rnce** *converter. If the HIBE scheme is IND-CCA secure with our additional assumptions, then we have*

$$\mathsf{R}^{\mathsf{ad\text{-}hibe}} \xrightarrow[\mathsf{asympt}]{\mathbf{rnce}\circ\mathbf{hibe}} \mathsf{S}^{\mathsf{ad\text{-}hibe}},$$

under the same event mapping as in Theorem 6.3.3.

Proof (Sketch). We sketch the simulator and simultaneously also argue why this simulation strategy makes the two worlds actually indistinguishable. Note that we focus here only on the RNCE parts, referring to the proof of Theorem 6.3.3, as to why the HIBE scheme provides proper healing, forward secrecy, and post-impersonation security.

The simulator essentially executes $\boldsymbol{\sigma}_{\mathsf{hibe}}$ (cf. Appendix C.2.2), except that it also internally samples all RNCE keys, including the trapdoors, and instead of encrypting a zero-string, it encrypts a fake RNCE ciphertext.

The randomness and public-key memory for RNCE are trivially simulatable. Furthermore, all memory and randomness resources of the HIBE construction are simulated as in $\boldsymbol{\sigma}_{\mathsf{hibe}}$. Clearly, we can also simulate both the `read` and `deliver` for the channels from Bob to Alice, where for instance *same* is determined analogous to $\boldsymbol{\sigma}_{\mathsf{hibe}}$ just also taking those public keys into account.

More interesting is the simulation of the channels from Alice to Bob (whose confidentiality the scheme protects) and the memories storing the corresponding RNCE secret keys. We consider three cases: (1) the parties are in sync (before an active impersonation) and the adversary simply forwards the message, (2) the parties are in sync and adversary tries to inject a message, and (3) the parties are already out of sync.

In sync. Consider one channel and the associated secret key, where the parties are in sync and the adversary does not carry out a successful impersonation.

On every `read` input to the channel, the simulator outputs either a proper encryption (if the message is known), or a HIBE-encryption of a fake ciphertext. For the key leakage, there are the following options:

- The distinguisher did not input the message to be encrypted under that key yet. The simulator outputs the honest secret key. Later on the `read` input to the channel, the message will be known to the simulator due to our can-leak predicate. Hence, the simulation is perfect.
- The message has been received. The simulator outputs $\perp$, since in the real world Bob would erase the key.

- A message is in transmission. Due to our can-leak predicate, this message is revealed to the adversary, so the simulator can produce a fake secret key that explains it. Indistinguishability follows from the security of RNCE.

For the `deliver` command, we proceed the same way as σ_{hibe}: we either forward the message (if it is the same ciphertext), or trigger an error (by our additional assumption on the HIBE scheme).

First injection. Consider now a channel, where parties are still in sync, and the adversary tries to deliver her own message. We proceed the same way as σ_{hibe} and just decrypt the ciphertext and observe the result, where the following RNCE is used:

- If the simulator already output a fake RCNE secret key, then it uses that one to decrypt.
- Otherwise, it uses the *honest* RNCE secret key. Here security follows from the CCA1-security of RNCE: a fake key and ciphertext are indistinguishable from the honest ones even given decryptions of adversarially chosen messages under the honest key before the fake key has been produced. Hence, even if the simulator later has to provide a fake RNCE key that explains Alice's message, this is indistinguishable from the real world.

Out of sync. Once the parties are out of sync, the adversary is allowed to learn the RNCE-keys without revealing the messages to the simulator (and thus allowing the simulator to output a fake secret key). The confidentiality of all channels after an impersonation attack is, however, guaranteed by the HIBE protocol, even if Bob's state is fully revealed to the adversary. Thus, the adversary never actually gets to see the fake RNCE ciphertexts (which are encrypted with the HIBE-scheme). As a consequence, the real RNCE keys can be safely revealed. □

6.5 Asynchronous Ratcheting as Continuous Key Agreement

Many secure messaging protocols, including Signal, proceed by combining some form of continuous key agreement (the asynchronous ratcheting layer) with a forward-secure symmetric encryption scheme (the synchronous ratcheting layer). Thus, the continuous key agreement primitive appears to be a natural abstraction boundary. Indeed, Alwen et al. [ACD19] modularize and abstract Signal in this manner by formalizing a notion of Continuous Key Agreement (CKA), which they then combine (with the help of a special PRF) with Forward-Secure AEAD (FS-AEAD).

In this section, we outline how such a CKA notion can be naturally expressed within our framework as a protocol that uses a given communication network to construct a sequence of keys (while preserving the communication network). We thereby focus on CKA only—the use of FS-AEAD to achieve secure messaging would then follow along the same lines as in [ACD19].

Continuous key agreement

We first briefly recall the CKA primitive, and refer to [ACD19] for further details. The setting is that of interlocked (or 'ping-pong') communication over authenticated channels, initiated by Alice. This means that in each odd round (which, following [ACD19], we call an *epoch*), Alice sends a message to Bob, and in each even round Bob sends a message to Alice. The goal is to provide a continuous stream of keys: each sending and receiving operation outputs a symmetric key (later used in the symmetric ratchet). This means that in each epoch (when a single message is sent and received) the parties produce a new key in the key stream (see Figure 6.8). Formally, a CKA scheme consists of four algorithms, of which $\mathsf{CKA.S}$ is randomized and the others deterministic:

$\mathsf{CKA.I\text{-}A}$ ($\mathsf{CKA.I\text{-}B}$)**:** On input a shared secret k, output the initial state $\gamma^{\mathtt{A}}$ of Alice (the initiator) ($\gamma^{\mathtt{B}}$ of Bob (the responder)).

$\mathsf{CKA.S}$**:** On input a sending state $\gamma^{\mathtt{A}}$, output a new receiving state, an update message T (sent to the partner), and the next shared key I.

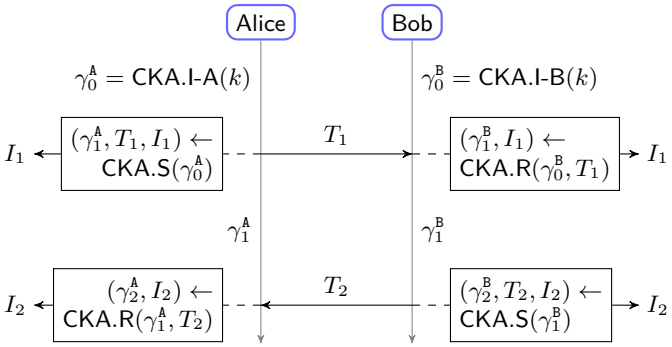


Figure 6.8: The synchronous execution of CKA, epochs 1 and 2. Alice sends in odd epochs, while Bob sends in even ones. In epoch 1, the parties produce the key I_1, which is available to Alice (the initiator in this key agreement) immediately, and to Bob (the responder) only after he receives the message T_1.

CKA.R: On input a receiving state γ^{B} and an update message T (received from the partner), output a new sending state and the next shared key I.

For correctness, we want that both parties produce the same key stream. The standard security properties include indisitnguishability: each key is indistinguishable from an (independent) random one, even given a number of other keys. Moreover, in case of a secret-state compromise, we require (1) forward secrecy: security of previously generated keys is not affected, and (2) healing: after a number of epochs since the last compromise, the security is restored. The scheme is parameterized by Δ, denoting the number of epochs that need to pass since epoch e until the state contains no information about the e-th key. We refer to [ACD19] for a formal description of the CKA security game.

The constructed key stream

In contrast of the previously presented constructions, the goal of a CKA protocol is not to enhance the guarantees of the channels (at least not directly). Rather, the goal is to construct a sequence of keys while preserving the channels and their respective guarantees.

Resource $\mathsf{Key}^{\mathsf{id},(\mathtt{I},\mathtt{R})}$

Parameters:

- Identity id, and interface names I (initiator) and R (responder)
- Predicates $\mathfrak{R}_{\mathtt{U}}$ (can be read by $\mathtt{U}$), $\mathfrak{L}_{\mathtt{U}}$ ($\mathtt{U}$'s copy can leak), and $\mathfrak{I}_{\mathtt{U}}$ (can be injected for $\mathtt{U}$), for $\mathtt{U} \in \{\mathsf{I}, \mathsf{R}\}$.

Events: $\mathcal{E}^{\mathsf{set},\mathtt{U}}$, $\mathcal{E}^{\mathsf{received},\mathtt{U}}$, $\mathcal{E}^{\mathsf{leaked}}$

Initialization

$k \leftarrow \mathcal{K}$
$k_{\mathsf{R}}, k_{\mathsf{I}} \leftarrow \mathtt{uninit}$
$inj_{\mathsf{R}}, inj_{\mathsf{I}} \leftarrow \bot$

Interface $\mathtt{E}$

Input: $(\mathtt{inject}, inj, \mathtt{U}) \in (2^{\mathcal{E}} \to \mathcal{K}) \times \{\mathtt{I}, \mathtt{R}\}$
if $k_{\mathtt{U}} \neq \mathtt{uninit}$ **then output** $\bot$
$inj_{\mathtt{U}} \leftarrow inj$
output $\mathtt{ok}$

Input: $(\mathtt{read}, \mathtt{U}) \in \{\mathtt{I}, \mathtt{R}\}$
if $\neg\mathfrak{L}_{\mathtt{U}}(\mathcal{E}) \vee k_{\mathtt{U}} = \mathtt{erased} \vee inj_{\mathtt{U}} \neq \bot$ **then**
  output $\bot$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}$
output k

Interface $\mathtt{U} \in \{\mathtt{I}, \mathtt{R}\}$

Input: $\mathtt{read}$
if $\neg\mathfrak{R}_{\mathtt{U}}(\mathcal{E}) \vee k_{\mathtt{U}} = \mathtt{erased}$ **then**
  output $\bot$
if $k_{\mathtt{U}} = \mathtt{uninit}$ **then**
  if $\mathfrak{I}_{\mathtt{U}}(\mathcal{E}) \wedge inj_{\mathtt{U}} \neq \bot$ **then**
    $k_{\mathtt{U}} \leftarrow inj_{\mathtt{U}}(\mathcal{E})$
    $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{set},\mathtt{U}}$
  else
    $k_{\mathtt{U}} \leftarrow k$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{received},\mathtt{U}}$
output $k_{\mathtt{U}}$

Input: $\mathtt{erase}$
$k_{\mathtt{U}} \leftarrow \mathtt{erased}$
output $\mathtt{ok}$

Figure 6.9: A formal description of the key resource. Note that the adversary can not only learn the key, but (if allowed) also set it individually for both parties. Note that for setting the key he is allowed to submit a function *inj* that determines the key (upon when the party first fetches it) based on the event history. That is, the party obtains $inj(\mathcal{E})$.

We model this sequence of keys as the parallel composition of many individual key resources $\mathsf{Key}^{e,(\mathtt{A},\mathtt{B})}$ shared between Alice and Bob, where e denotes the epoch number. A formal definition of the key resource is given in Figure 6.9. On a high level, it essentially behaves like a pair of memory resources that have the respective key stored. More concretely, the key's interfaces and their capabilities are as follows:

- Parties can read the key once it is available to them. The avail-

ability is determined via the respective can-read predicates $\mathfrak{R}_{\mathsf{I}}$ and $\mathfrak{R}_{\mathsf{R}}$. Moreover, the parties can each request to securely erase their respective copy of the key.

- The adversary $\mathtt{E}$ can potentially leak the key from either party, if this is allowed by the respective can-leak predicate $\mathfrak{L}_{\mathtt{U}}$, and the key has not been securely erased yet. Note that the leaked key still looks random.

- Moreover, in certain circumstances the key is no longer random (e.g., if the secret state used to produce it leaks). This corresponds to the adversary being able to inject her own key. Specifically, the adversary can inject, for each party, a function *inj*, that is invoked when a party fetches their key and the can-inject predicate evaluates to true. The function computes the injected key, given the current event history.

The setting

Although CKA is a primitive that mandates an interlocked communication pattern, it is still intended to be used on top of asynchronous communication between Alice and Bob. In particular, a party will attach the latest synchronous update message T to each asynchronous message it sends. This way, the CKA protocol proceeds as fast as possible: as soon as at least one of the messages carrying a new, previously unseen update arrives, a party can move to the next epoch. For this reason, the real world involves the usual sequence of (insecure) channels between Alice and Bob:

$$\mathsf{Channels} := \left[\left\{\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}\right\}_{i\in[n]}, \left\{\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}\right\}_{j\in[n]}\right].$$

In the real world, we have various memory resources: Each user $\mathtt{U}$ stores his current epoch number e in the insecure memory $\mathsf{IMem}^{ep,\mathtt{U}}$, and he stores his CKA state corresponding to epoch e in $\mathsf{Mem}^{st_e,\mathtt{U}}$. Moreover, he stores the key I_e produced in epoch e in $\mathsf{OMem}^{key_e,\mathtt{U}}$ (where OMem denotes an observable memory that behaves like Mem, except that it triggers an $\mathcal{E}^{\mathsf{read}}$ event when the honest reader accesses the content for

the first time[8]). Overall, we have

$$\mathsf{Mems} \coloneqq \Big[\big\{\mathsf{Mem}^{st_e,\mathtt{U}}, \mathsf{OMem}^{key_e,\mathtt{U}},\big\}_{e\in[2n],\mathtt{U}\in\{\mathtt{A},\mathtt{B}\}}, \big\{\mathsf{IMem}^{ep,\mathtt{U}}\big\}_{\mathtt{U}\in\{\mathtt{A},\mathtt{B}\}}\Big],$$

Furthermore, there are randomness resources $\mathsf{Rand}^{e,\mathtt{U}}$ for the CKA.S operation:

$$\mathsf{Rand} \coloneqq \Big[\big\{\mathsf{Rand}^{e,\mathtt{A}}\big\}_{odd\ e\in[2n]}, \big\{\mathsf{Rand}^{e,\mathtt{B}}\big\}_{even\ e\in[2n]}\Big].$$

Finally, CKA requires setup in the form of a shared key, which is used by Alice and Bob to derive their initial states via CKA.I-A and CKA.I-B, respectively. Security of this operation is, however, outside the scope of CKA (in particular, the shared key is never revealed to the adversary). We model this by a real-world setup resource $\mathsf{CKA\text{-}Setup}^{(\mathtt{I},\mathtt{R})}$, formally defined in Figure 6.10, that executes the initialization procedure and provides the initial states $\gamma_0^{\mathtt{A}}$ and $\gamma_0^{\mathtt{B}}$ to the respective parties. That is, analogous to our key-resource, the setup resource essentially corresponds to a pair of memories $[\mathsf{Mem}^{st_0,\mathtt{A}}, \mathsf{Mem}^{st_0,\mathtt{B}}]$, except that instead of the parties writing to it, their states are "magically" initialized.

Putting it all together, we have the following assumed resources:

$$\mathsf{R}_{\mathsf{CKA}} \coloneqq \Big[\mathsf{CKA\text{-}Setup}^{(\mathtt{A},\mathtt{B})}, \mathsf{Channels}, \mathsf{Mems}, \mathsf{Rand}\Big], \tag{6.11}$$

to which the converters implementing the CKA protocol will be attached.

The protocol

As we already hinted before, the protocol works by attaching to each message sent on one of the channels the next CKA message T. That is, we execute the protocol sketched on Figure 6.8, except each message T_e is repeated with each asynchronous message.

Figure 6.11 formally describes the protocol executed by Alice. Her state in epoch e is stored in $\mathsf{Mem}^{st_e,\mathtt{A}}$. For an even e, this is a 'receiving' state (recall that Alice can only receive messages in such epochs), that contains simply the CKA state γ. For an odd e, this is a 'sending' state, containing the pair (γ, T), where T is the current update message,

[8] We will need this to specify our interval-wise guarantee to avoid the commitment problem in case an ideal key has been output.

Resource CKA-Setup$^{(\mathtt{I},\mathtt{R})}$

Parameters: interfaces I (initiator) and R (responder), can-leak predicate $\mathfrak{L}$.

Events: $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(st_0,\mathsf{I})}, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(st_0,\mathsf{R})}$

Initialization

$k \leftarrow\!\!\!\!\!\!\$\ \mathcal{K}$
$\gamma^{\mathsf{A}} \leftarrow \mathsf{CKA.I\text{-}A}(k); \gamma^{\mathsf{B}} \leftarrow \mathsf{CKA.I\text{-}B}(k)$

Interface E

Input: $(\mathsf{read}, \mathsf{U}) \in \{\mathsf{I}, \mathsf{R}\}$
if $\neg\mathfrak{L}(\mathcal{E})$ **then output** $\bot$
if $\gamma^{\mathsf{U}} \neq \bot$ **then** $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(st_0,\mathsf{U})}$
output γ^{U}

Interface $\mathsf{U} \in \{\mathtt{I}, \mathtt{R}\}$

Input: `read`
output γ^{U}

Input: `erase`
$\gamma^{\mathsf{U}} \leftarrow \bot$
output `ok`

Figure 6.10: The resource encoding the setup and initialization of CKA.

attached to all Alice's messages in this epoch. On each `write` input on the outside interface of a channel, the protocol first decides whether this message initiates a new epoch (if e is even) or not (if e is odd). If the protocol stays in the same epoch, then it fetches the 'sending' state (γ, T) from $\mathsf{Mem}^{st_e,\mathtt{A}}$ and simply attaches T to the message. Otherwise, the protocol uses the 'receiving' state fetched from the memory for the previous (even) epoch to compute the update T. This latter operation produces, as a byproduct, a new key I, which is stored in $\mathsf{OMem}^{key_e,\mathtt{A}}$. A `read` input is processed analogously. First, the message and the update T are read from the real channel. If T has not been seen before, it is used to initialize $\mathsf{Mem}^{st_e,\mathtt{A}}$, for the new epoch e, with a new receiving state. On input `read` on the outside interface of a key, the key is retrieved from $\mathsf{Mem}^{key_e,\mathtt{A}}$.

Bob's protocol essentially works analogously—swapping odd and even epochs. There are some minor differences with respect to initialization that reflect the fact that overall Alice acts as initiator and Bob as responder, i.e., Bob cannot send a message before having received one. We omit a formal specification of the respective converter cka-responder, since the necessary modifications should be self explanatory.

Converter cka-initiator

Emulating Int. A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}, i \in [n]$

Input: $(\mathtt{send}, m, ad)$
  require only called once & isAvailable
  call $e \leftarrow$ `read` at int. A of $\mathsf{IMem}^{ep,\mathtt{A}}$
  if $e = \bot \vee e \bmod 2 = 0$ **then**
    if $e = \bot$ **then**
      call $\gamma \leftarrow$ `read` at interface A of $\mathsf{CKA\text{-}Setup}^{(\mathtt{A},\mathtt{B})}$
      call `erase` at int. A of $\mathsf{CKA\text{-}Setup}^{(\mathtt{A},\mathtt{B})}$
      $e \leftarrow 1$
    else
      call $\gamma \leftarrow$ `read` at int. A of $\mathsf{Mem}^{st_e,\mathtt{A}}$
      call `erase` at int. A of $\mathsf{Mem}^{st_e,\mathtt{A}}$
      $e \leftarrow e + 1$
    call $r \leftarrow$ `sample` at int. A of $\mathsf{Rand}^{e,\mathtt{A}}$
    $(\gamma, T, I) \leftarrow \mathsf{CKA.S}(\gamma; r)$
    call $(\mathtt{write}, e)$ at int. A of $\mathsf{IMem}^{ep,\mathtt{A}}$
    call $(\mathtt{write}, \gamma, T)$ at int. A of $\mathsf{Mem}^{st_e,\mathtt{A}}$
    call $(\mathtt{write}, I)$ at int. A of $\mathsf{OMem}^{key_e,\mathtt{A}}$
  else
    call $(\gamma, T) \leftarrow$ `read` at int. A of $\mathsf{Mem}^{st_e,\mathtt{A}}$
  call $succ \leftarrow (\mathtt{send}, m, (ad, e, T))$ at interface A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  return $succ$

Input: `isAvailable`
  call $succ \leftarrow$ `isAvailable` at int. A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  return $succ$

Emulating Int. A of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}, j \in [n]$

Input: `receive`
  require only called once & isAvailable
  call $e \leftarrow$ `read` at int. A of $\mathsf{IMem}^{ep,\mathtt{A}}$
  call $(m, (ad, e', T')) \leftarrow$ `receive` at interface A of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  if $\neg(2 \le e' \le e + 1) \vee e' \bmod 2 = 1$ **then**
    return $\bot$
  else if $e' = e + 1$ **then**
    call $(\gamma, T) \leftarrow$ `read` at int. A of $\mathsf{Mem}^{st_e,\mathtt{A}}$
    $(\gamma, I) \leftarrow \mathsf{CKA.R}(\gamma, T')$
    $e \leftarrow e + 1$
    call $(\mathtt{write}, e)$ at int. A of $\mathsf{IMem}^{ep,\mathtt{A}}$
    call `erase` at int. A of $\mathsf{Mem}^{st_{e-1},\mathtt{A}}$
    call $(\mathtt{write}, \gamma)$ at int. A of $\mathsf{Mem}^{st_e,\mathtt{A}}$
    call $(\mathtt{write}, I)$ at int. A of $\mathsf{OMem}^{key_e,\mathtt{A}}$
  return (m, ad)

Input: `isAvailable`
  call $e \leftarrow$ `read` at int. A of $\mathsf{IMem}^{ep,\mathtt{A}}$
  call $succ \leftarrow$ `isAvailable` at int. A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  return $e \neq \bot \wedge succ$

Emulating Int. A of $\mathsf{Key}^{e,(\mathtt{A},\mathtt{B})}, e \in [2n]$

Input: `read`
  call $k \leftarrow$ `read` at int. A of $\mathsf{OMem}^{key_e,\mathtt{A}}$
  return k

Input: `erase`
  call `erase` at int. A of $\mathsf{OMem}^{key_e,\mathtt{A}}$
  return `ok`

Figure 6.11: The protocol cka-initiator implementing CKA, executed by the initiator, Alice. The protocol cka-responder executed by the responder, Bob, is analogous.

The construction

As mentioned before, the goal of the CKA protocol is to construct a sequence of keys while preserving the channels and their respective guarantees. Thus, the ideal world consist of channels and the (potentially) constructed key resources:

$$\mathsf{S}_{\mathsf{CKA}} \coloneqq \left[\left\{\mathsf{Key}^{e,(\mathtt{A},\mathtt{B})}\right\}_{e\in[2n]}, \mathsf{Channels}\right], \tag{6.12}$$

where $2n$ is an upper bound of the number of epochs that can be initiated when the parties send at most n messages each. If fewer epochs occur, then we model this by simply *not enabling* the corresponding key resources, i.e, the resources formally exist but are not available to the parties.

Let us now describe the properties of the constructed key sequence, as determined by the predicates can-read, can-leak, and can-set. Both parties can read the key $\mathsf{Key}^{e,(\mathtt{A},\mathtt{B})}$ as soon as they entered the e-th epoch:

$$\mathfrak{R}^{\mathsf{S}_{\mathsf{CKA}}}_{\mathtt{U},\mathsf{Key}(e,(A,B))}(\mathcal{E}) \coloneqq \mathcal{E}^{\mathsf{in\text{-}epoch}}_{e,\mathtt{U}}, \tag{6.13}$$

where $\mathcal{E}^{\mathsf{in\text{-}epoch}}_{e,\mathtt{U}}$ denotes the event that user $\mathtt{U}$ entered the e-th epoch. Note that as long as Eve does not inject a forgery, this can easily be determined from the event history that contains the sequence of all sent and received messages and their respective order. Once Eve injects, we can, for the sake of those predicates, assume that she will always advance the session (cf. the simulator in the proof). Hence, the event $\mathcal{E}^{\mathsf{in\text{-}epoch}}_{e,\mathtt{U}}$ can easily be made formal.

The can-leak predicate of a key corresponds to the can-leak predicate of the memory storing it, that is it leaks to the adversary if the memory leaks:

$$\mathfrak{L}^{\mathsf{S}_{\mathsf{CKA}}}_{\mathtt{U},\mathsf{Key}(e,(A,B))}(\mathcal{E}) \coloneqq \mathfrak{L}^{\mathsf{R}_{\mathsf{CKA}}}_{\mathtt{U},\mathsf{Mem}(key_e,\mathtt{U})}(\tilde{\mathcal{E}}), \tag{6.14}$$

which captures the situations where key directly leaks.

The stronger corruption, where Eve sets the key, is controlled by the can-set predicate. This models situations, in which a key is no longer random (and hence can be set by Eve). Roughly, keys are not random in two situations: First, a key for epoch e is not random if either the randomness to generate it leaked, or a user's state leaks in any epoch $e' \in \{e-1,\ldots,e+\Delta-1\}$. Note that this is just a safe approximation.

For instance, while the scheme could heal for $\mathtt{U}$ between epoch $e-1$ and e (if e is a sending epoch for him) or not (otherwise), it is guaranteed to heal between epoch $e-2$ and e, and analogous for $e+\Delta$. Since the can-set predicate is only defined on past events, for now we only say that it is true if a user's state leaked in epochs $e-1$ or e. The other epochs $e+1,\ldots,e+\Delta-1$ are considered when we deal with the commitment problem. As a consequence, we define the following composed event:

$$\mathcal{E}^{\mathsf{state\text{-}leaked}}_{\mathtt{U},e} := \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(st_e,\mathtt{U})} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(st_{e-1},\mathtt{U})} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(e,\mathtt{U})}.$$

Second, all keys for epochs following an active attack are not random either (we give up on any guarantees in such case; originally, CKA does not consider injections at all). Specifically, assume that the adversary injects the i-th message. Clearly, she can influence the receiver's key of the corresponding epoch e_i. In addition, we allow the adversary to set the keys of all upcoming epochs e, and thus give up all guarantees for epochs e with $e_i \le e$, as expressed by the following event:

$$\mathcal{E}^{\mathsf{injected\text{-}before}}_{\mathtt{U}\to\bar{\mathtt{U}},e} := \exists i\ \exists e_i \le e : \big(\mathcal{E}^{\mathsf{in\text{-}epoch}}_{e_i-1,\mathtt{U}} \prec \mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i,\mathtt{U}\to\bar{\mathtt{U}})} \prec \mathcal{E}^{\mathsf{in\text{-}epoch}}_{e_i+1,\mathtt{U}}\big) \wedge \mathcal{E}^{\mathsf{injected}}_{\mathsf{Ch}(i,\mathtt{U}\to\bar{\mathtt{U}})},$$

where the first condition just asserts that e_i is indeed the i-th message's epoch.

Finally, for the epoch of the first injection, CKA does not guarantee the randomness of neither the receiver's nor the sender's key (by injecting the modified sender's message, Eve can cause the keys to be correlated).

Overall, we define the following can-inject predicates, which are fully symmetric for both parties:

$$\mathfrak{I}^{\mathsf{S_{CKA}}}_{\mathtt{U},\mathsf{Key}(e,(A,B))}(\mathcal{E}) := \mathcal{E}^{\mathsf{key\text{-}exposed}}_{\mathtt{A},e} \vee \mathcal{E}^{\mathsf{key\text{-}exposed}}_{\mathtt{B},e}, \tag{6.15}$$

where

$$\mathcal{E}^{\mathsf{key\text{-}exposed}}_{\mathtt{U},e} := \mathcal{E}^{\mathsf{state\text{-}leaked}}_{\mathtt{U},e} \vee \mathcal{E}^{\mathsf{injected\text{-}before}}_{\bar{\mathtt{U}}\to\mathtt{U},e}.$$

The commitment problem

The ideal world $\mathsf{S_{CKA}}$ with the predicates described above is trivially distinguishable from the real world. This is because, as already mentioned, leaking a state in epoch $e_l \in \{e+1,\ldots,e+\Delta-1\}$ compromises

the secrecy of the key from a past epoch e. However, in the adaptive setting, the can-set predicate for this key cannot take into account the future event of memory leakage.

As a consequence, we again turn towards interval-wise guarantees. That is, analogous to the game-based definitions of CKA, we only make a statement until the distinguisher decides to corrupt a memory that cannot be simulated. More concretely, if the distinguisher decides to "see" the ideal key in epoch e, then the memory cannot leak in epochs $e+1, \ldots, e+\Delta-1$. Overall, we define the event that indicates a trivial distinguishing attack as follows.

$$\mathcal{E}^{\mathsf{inval\text{-}leak}}_{e,\mathtt{U}} := \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(st_e,\mathtt{U})} \wedge \neg\exists \mathtt{U}', e' : \big(e-\Delta < e' \leq e\big) \\ \wedge \big(\mathcal{E}^{\mathsf{committed}}_{\mathtt{U}',e'} \prec \mathcal{E}^{\mathsf{key\text{-}exposed}}_{\mathtt{U}',e'}\big),$$

where

$$\mathcal{E}^{\mathsf{committed}}_{\mathtt{U},e} := \mathcal{E}^{\mathsf{leaked}}_{\mathsf{OMem}(key_e,\mathtt{U})} \vee \mathcal{E}^{\mathsf{read}}_{\mathsf{OMem}(key_e,\mathtt{U})}$$

simply denotes the event that the key has either been output to the user or leaked to the adversary.

Moreover, the commitment problem also occurs for active injections: if the adversary chooses to change the value T to T' for the first time, then this also compromises the sender's key, which at this time might already have been used. Analogous to the CKA security game of [ACD19], we thus cannot simulate past injections if the sender is committed on a uniform key for which the receiver did not yet receive the message. Hence, we introduce the following indicator event:

$$\mathcal{E}^{\mathsf{inval\text{-}inject}}_{i,\mathtt{U}} := \mathcal{E}^{\mathsf{received}(\mathtt{false})}_{\mathsf{Ch}(i,\mathtt{U}\to\bar{\mathtt{U}})} \\ \wedge \exists e : \neg\mathcal{E}^{\mathsf{in\text{-}epoch}}_{e,\bar{\mathtt{U}}} \ \wedge \ \mathcal{E}^{\mathsf{committed}}_{\mathtt{U},e} \prec \mathcal{E}^{\mathsf{key\text{-}exposed}}_{\mathtt{U},e}.$$

Note that this can only occur for the very first injection, as it immediately triggers $\mathcal{E}^{\mathsf{key\text{-}exposed}}_{\mathtt{U},e_i}$ for all future epochs.

In summary, we can thus simulate until the following event

$$\mathcal{E}^{\mathsf{com\text{-}prob}} := \bigvee_{\mathtt{U}\in\{\mathtt{A},\mathtt{B}\}} \Big(\bigvee_{e\in[2n]} \mathcal{E}^{\mathsf{inval\text{-}leak}}_{e,\mathtt{U}} \vee \bigvee_{i\in[n]} \mathcal{E}^{\mathsf{inval\text{-}inject}}_{i,\mathtt{U}} \Big), \tag{6.16}$$

occurs, indicating a trivial win for the distinguisher (after which we are not interested in giving guarantees).

Summary and analysis

With this workaround for the commitment problem in place, the CKA scheme now achieves the described construction, as summarized in the following theorem.

Theorem 6.5.1. *If the CKA scheme* CKA *is* (Δ, ϵ)*-secure for some negligible* ϵ*, then if we map each event* $\mathcal{E}^{\mathsf{received},\mathtt{U}}_{\mathsf{Key}(e,(\mathtt{A},\mathtt{B}))}$ *to* $\mathcal{E}^{\mathsf{read}}_{\mathsf{OMem}(key_e,\mathtt{U})}$ *and each event* $\mathcal{E}^{\mathsf{leaked},\mathtt{U}}_{\mathsf{Key}(e,(\mathtt{A},\mathtt{B}))}$ *to* $\mathcal{E}^{\mathsf{leaked}}_{\mathsf{OMem}(key_e,\mathtt{U})}$*, we have*

$$\mathsf{R}_{\mathsf{CKA}} \xrightarrow[\text{interval-asym}]{\boldsymbol{cka},[\mathtt{true},\mathcal{E}^{\text{com-prob}}]} \mathsf{S}_{\mathsf{CKA}},$$

for the protocol $\boldsymbol{cka} := (\mathsf{cka\text{-}initiator}, \mathsf{cka\text{-}responder})$*, the resources* $\mathsf{R}_{\mathsf{CKA}}$ *and* $\mathsf{S}_{\mathsf{CKA}}$ *are as defined in Equations* (6.11) *and* (6.12) *with the predicates as defined in Equations* (6.13) *to* (6.15)*, and the event* $\mathcal{E}^{\mathsf{com\text{-}prob}}$ *defined in Equation* (6.16)*.*

Proof. The simulator $\boldsymbol{\sigma}_{\mathsf{CKA}}$ (cf. Appendix C.3.1 for a formal description) has to emulate the state memories and randomness resources. Moreover, it has to adjust the channel interfaces (where the CKA-values (e, T) are not present in the ideal world) and map the key resources to the corresponding memory resources in the real world. Observe that the simulator sees all values (e, T) that get delivered to either party as the adversary needs to specify them as part of the associated data. It stores them in the transcript Tr, i.e, sets $\mathsf{Tr}[i, B] \leftarrow (e, T)$ when the message on the i-th channel to Bob contains those values. Given the event history, the simulator moreover knows which messages have been delivered, and thus can always compute the epoch a party is in.

Consider first the initial "secure" execution where the adversary only forwards messages. The simulator samples all the randomness and pre-computes the real-world states, keys and update messages for all epochs upfront. With this and the knowledge of which epoch the parties are in, he can trivially simulate the insecure memories storing the epoch number, the randomness resources, the leakage of the channels (containing (e, T)), and the state memories — at least as long as no injection has happened.

Upon a key-memory corruption, he leaks the value stored in the corresponding ideal-world key resource. It remains to show that the

parties' can-inject predicates permit that (1) the "ideal" key is indistinguishable from the real key I_e if the key resource does not permit programming, or (2) he can program it consistently. Moreover, we also still need to argue that an active injection is handled properly. To this end, we consider the three stages: while the parties are still in sync, the first injection, and the phase where the parties are out of sync.

In sync. Consider the key for the e-th epoch. The CKA security game ensures that this is indistinguishable from a uniform random key unless a user's state leaks in any epoch $e' \in \{e-1, \ldots, e+\Delta-1\}$, or the randomness to generate it leaked. The can-inject predicate of the key resource ensures that the simulator is allowed to program the key if a user's state of an epoch $e' \in \{e-1, e\}$, or the randomness, leaked. The interval-wise guarantee moreover ensures that we only have to simulate leakage of the state of the epochs $e' \in \{e+1, \ldots, e+\Delta-1\}$ if the key has been exposed in any way.

Hence, it suffices that the simulator always just tried to program the ideal-world keys to the precomputed ones (which are the same as in the real-world). In particular, every time a message for epoch e (as determined by the public associated data) is delivered to Bob by the adversary (via either `fwd` or `dlv`), $\boldsymbol{\sigma}_{\mathsf{CKA}}$ updates the inj_{B} function of the e-th key. The function is defined as $inj_{\mathsf{B}}(\mathsf{Tr}, e, \gamma^{\mathsf{A}}, \gamma^{\mathsf{B}}, I, \mathsf{B}, \cdot)$, where the last argument is the event history, provided by the key resource at the time the key is fetched, and the other arguments contain the whole state of the simulation. Given this, the function computes the key Bob would fetch in the real world.

First injection. Consider now a channel, where parties are still in sync, and the adversary tries to deliver her own message, including her own value T' for epoch e'. First observe that if in the ideal world, the simulator is already committed on any ideal key, especially the one for epoch e', then the injection is not allowed.

In any case, the simulator just reruns the CKA protocol from e' on. This, among others, ensures that the leakage of the channels and the state memories keeps being consistent even once the parties are out of sync. Furthermore, using the newly obtained key, the

simulator programs the receiver' key as above. Note that he already programmed the sender' key consistently, which is after the injection also the one the key resource will output by the definition of the can-inject predicate.

Out of sync. Once the parties are out of sync, we give up on any guarantee. Hence, for any delivery attempt (e', T'), the simulator can rerun the protocol from epoch e' on and consistently program all the keys. □

Chapter 7

Conclusion

This thesis is driven by one question: How can we overcome certain obstacles that hinder the adoption of composable security definitions to become the standard in cryptography? We ultimately believe that every meaningful security statement can be phrased composably, i.e., such that it is clear how one can build on it and such that it either holds in any arbitrary environment, or at least makes the restrictions crystal clear as part of the statement. The key to success in this endeavor is finding the right type of composable statements, including the aspect of finding the right mathematical type of object (i.e., abstraction boundary) as well as finding the right types of statements to make about them. The Constructive Cryptography framework is, thus, the perfect match for this research. Its top-down abstraction and unique approach of understanding security statements as a special instance of specification abstraction, consolidated in an overarching definitional framework with a supporting theory, clear guiding rules, and yet plenty of flexibility, together provide an excellent foundation to explore novel answers to those questions.

In Chapter 3, we propose a novel type of mathematical object to consider in Constructive Cryptography. Namely, we extend the mathematical objects with the notion of global events to enhance the flexibility and fine-granularity of the abstraction boundary that helps to enhance modularity and phrase statements in a minimal manner. We demonstrate its wide applicability in Chapters 5 and 6. In Chapter 4, we

introduce the notion of context-restricted constructions, that allow to explicitly formalize statements that do not hold in arbitrary contexts but only within a well-specified set, while providing the expected remaining composition guarantees. In particular, we show that such an approach can provide explicit semantics to multi-stage game-based notions. In Chapter 5, we illustrate that composable statements about schemes for which provably no UC-style simulation-based statement exist are not a paradox, if one frees oneself from the overly restrictive understanding of composable security being equivalent to the existence of a simulator. In contrast, we show how such protocols' achieved guarantees can be naturally formalized using our novel type of interval-wise specifications. Finally, in Chapter 6 we demonstrate the applicability of our generalizations to secure messaging; an area notoriously known for complicated and overly specific game-base security definitions, especially resulting in poor transferability of results among different works.

We hope that this thesis helps to spark new research interest in the field of composable security. In light of this thesis, we encourage to reconsider cryptographic schemes and protocols, which so far were only known to satisfy standalone game-based security notions, and aim at phrasing their guarantees as constructions of resource specifications. In particular, multi-stage security games without clear executional semantics might benefit from a composable clean slate approach. Finally, in the realm of the simulator commitment problem, for instance the seminal result in the context of MPC by Canetti, Feige, Goldreich, and Naor [CFGN96], indicating that a protocol designed in the secure-channel model cannot be generically instantiated using public-key cryptography unless non-committing encryption is employed, deserves reconsideration.

Appendix A

Details of Chapter 4

A.1 Proof of Lemma 4.5.8

Lemma 4.5.8. *Let $k' \coloneqq \min(k, \log|H.\mathcal{Y}|)$. If H is $\mathcal{C}_p^{\mathsf{s\text{-}splt}} \cap \mathcal{C}_{n,k'}^{\mathsf{s-me}}$ RO-CRI secure, then H is also $\mathcal{C}_{p,r}^{\mathsf{r\text{-}splt}} \cap \mathcal{C}_{n,k}^{\mathsf{s-me}}$ RO-CRI secure.*

More concretely, let $\mathcal{D}$ denote the set of distinguishers. Then there exists a translation of the distinguisher $\rho\colon \mathcal{D} \times \left(\mathcal{C}_{p,r}^{\mathsf{r\text{-}splt}} \cap \mathcal{C}_{n,k}^{\mathsf{s-me}}\right) \to \mathcal{D}$ and a translation of the context $\psi\colon \mathcal{C}_{p,r}^{\mathsf{r\text{-}splt}} \cap \mathcal{C}_{n,k}^{\mathsf{s-me}} \to \mathcal{C}_p^{\mathsf{s\text{-}splt}} \cap \mathcal{C}_{n,k'}^{\mathsf{s-me}}$, such that for every $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}_{p,r}^{\mathsf{r\text{-}splt}} \cap \mathcal{C}_{n,k}^{\mathsf{s-me}}$ we have

$$\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) \leq \binom{npr}{2} 2^{-(k'-1)} + r \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathsf{f}',\mathsf{X}',\sigma}(\mathsf{D}')$$

with $\mathsf{D}' \coloneqq \rho(\mathsf{D}, \mathbf{f}, \mathsf{P})$ and $(\mathsf{f}', \mathsf{X}') \coloneqq \psi(\mathbf{f}, \mathsf{P})$.

Proof. Let $(\mathbf{f}, \mathsf{P}) \in \mathcal{C}_{p,r}^{\mathsf{r\text{-}splt}} \cap \mathcal{C}_{n,k}^{\mathsf{s-me}}$. By definition, we then have $\mathbf{f} \coloneqq \mathsf{g} \circ \mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}$ for some filter g. This filter can also be thought of as an reduction of the distinguisher (which follows from the composition-order independence [MR11]), and thus we can rewrite

$$\begin{aligned}\mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathbf{f},\mathsf{P},\sigma}(\mathsf{D}) &\coloneqq \Delta^{\mathsf{D}}(\mathbf{f}[\mathsf{H},\mathsf{P}], \mathbf{f}[\mathsf{RO},\mathsf{P}]\boldsymbol{\sigma}) \\ &= \Delta^{\mathsf{D}''}(\mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}[\mathsf{H},\mathsf{P}], \mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}[\mathsf{RO},\mathsf{P}]\boldsymbol{\sigma})\end{aligned}$$

with $\mathsf{D}'' = \rho_1(\mathsf{D}) \coloneqq \mathsf{D}\mathsf{g}$.

Consider the beacon resource B that has the same interface as the random oracle interface, but response with a fresh random value for each query (i.e., it ignores the consistency condition for repeated queries). Moreover, we introduce the following shorthand notation: $\mathsf{S}^{\mathsf{H}} \coloneqq \mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}[\mathsf{H},\mathsf{P}]$, $\mathsf{S}^{\mathsf{RO}} \coloneqq \mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}[\mathsf{RO},\mathsf{P}]\boldsymbol{\sigma}$, and $\mathsf{S}^{\mathsf{B}} \coloneqq \mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}[\mathsf{B},\mathsf{P}]\boldsymbol{\sigma}$, which allows the advantage of the distinguisher D'' to be expressed as

$$\Delta^{\mathsf{D}''}(\mathbf{f}[\mathsf{H},\mathsf{P}],\mathbf{f}[\mathsf{RO},\mathsf{P}]\boldsymbol{\sigma}) = \Delta^{\mathsf{D}''}(\mathsf{S}^{\mathsf{H}},\mathsf{S}^{\mathsf{B}}) + \Delta^{\mathsf{D}''}(\mathsf{S}^{\mathsf{B}},\mathsf{S}^{\mathsf{RO}}).$$

We now describe the reduction ρ_2 that bounds the first term of the sum with $\binom{npr}{2}2^{-k'} + r \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathsf{f}',\mathsf{X}',\sigma}(\mathsf{D}')$ using a simple hybrid argument.

Let $\{\mathsf{X}_\mathsf{i}\}_{i\in[q]}$ denote the sequence of hybrid resources that behave as follows: at the interface $\mathtt{E}$, the resource first outputs the index i and subsequently behaves exactly as P. At the interface $\mathtt{A}$, if $i = 1$ then it behaves exactly as P, and if $i > 1$ then it outputs n independent uniformly at random chosen values from the set $H.\mathcal{Y}$. It is easy to see, that if $\mathsf{P} \in \Theta^{\mathsf{s-me}}_{n,k'}$, then $\mathsf{X}_\mathsf{i} \in \Theta^{\mathsf{s-me}}_{n,k'}$ for all i. In addition, let X' denote the resource which chooses $i \in [q]$ uniformly at random and then behaves like X_i. Furthermore, let $\mathsf{f}' \coloneqq \mathbf{f}_p^{\mathsf{s\text{-}splt}}$ and, hence $(\mathsf{f}',\mathsf{X}') \in \mathcal{C}_p^{\mathsf{s\text{-}splt}} \cap \mathcal{C}_{n,k'}^{\mathsf{s-me}}$. Analogously to above, let us define the following shorthand notation: $\mathsf{T}^{\mathsf{H}} \coloneqq \mathbf{f}_p^{\mathsf{s\text{-}splt}}[\mathsf{R},\mathsf{X}']$, $\mathsf{T}^{\mathsf{H}}_\mathsf{i} \coloneqq \mathbf{f}_p^{\mathsf{s\text{-}splt}}[\mathsf{R},\mathsf{X}_\mathsf{i}]$, and $\mathsf{T}^{\mathsf{R}} \coloneqq \mathbf{f}_p^{\mathsf{s\text{-}splt}}[\mathsf{R},\mathsf{X}']\boldsymbol{\sigma}$ and $\mathsf{T}^{\mathsf{R}}_\mathsf{i} \coloneqq \mathbf{f}_p^{\mathsf{s\text{-}splt}}[\mathsf{R},\mathsf{X}_\mathsf{i}]\boldsymbol{\sigma}$ for $\mathsf{R} \in \{\mathsf{RO},\mathsf{B}\}$.

Now, consider the reduction $\mathsf{D}' \coloneqq \rho_2(\mathsf{D}'') = \rho_2(\rho_1(\mathsf{D}))$ where ρ_2 is implemented using a special type of system C that translates one setting into the other. Formally C is a converter that has an inside and an outside interface, where the inside interface connects to all the (merged) interfaces of the attached resource (here interface $\mathtt{A}$ and $\mathtt{E}$) and the outside interface becomes the interfaces of the composed resource. Now consider the following reduction system C, which on the inside expects to be connected either to the resource $\mathsf{T}^{\mathsf{H}}_\mathsf{i}$ or $\mathsf{T}^{\mathsf{B}}_\mathsf{i}$. At the outside interfaces, it simulates the according interfaces of $\mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}[\mathsf{H},\mathsf{P}]$ and $\mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}[\mathsf{B},\mathsf{P}]\boldsymbol{\sigma}$. The system C first gets the index i and the hash key hk at the inside interface. In every sequence of queries of the form $(\mathtt{query}, f_1), (\mathtt{repeat}, f_2), (\mathtt{repeat}, f_3), \ldots$, the queries 1 to $i-1$ are simulated internally as queries to the beacon by sampling a value uniformly at random and storing it in a buffer b. The i-th query in each such sequence is then answered using the actual resource connected at

the inside interface. For the remaining queries, the system C computes the hash function H itself. A formal description of the reduction system is provided in Figure A.1.

Resource C

Initialization

$b \leftarrow \bot^p$
$j \leftarrow 0$
call $hk \leftarrow$ `retrieve` at int. E of H
call $i \leftarrow$ `retrieve` at int. E of X

Emulating Interface `A` of $\mathbf{f}^{\mathsf{r\text{-}splt}}_{p,r}$

Input: `get`
$b' \leftarrow b$
$b \leftarrow \bot^p$
output b'

Input: $(\texttt{query}, f_1, \ldots, f_p)$
$j \leftarrow 1$
if $i = 1$ **then**
  call $b \leftarrow (\texttt{retrieve}, f_1, \ldots, f_p)$ at int. A of $\mathbf{f}^{\mathsf{s\text{-}splt}}_p$
else
  $b \twoheadleftarrow H.\mathcal{Y}^p$

Input: $(\texttt{repeat}, f_1, \ldots, f_p)$
$j \leftarrow j + 1$
if $b \neq \bot^p$ **then**
  for $\ell = 1, \ldots, p$ **do**
    if $j < i$ **then**
      $b[\ell] \twoheadleftarrow H.\mathcal{Y}$
    else if $j = i$ **then**
      call $b[\ell] \leftarrow (\texttt{retrieve}, (x, a) \mapsto f_\ell(x))$ at int. A of $\mathbf{f}^{\mathsf{s\text{-}splt}}_p$
    else
      $b[\ell] \leftarrow H(hk, f_l(b[\ell]))$

Emulating Interface `E` of i, $i \in \{H, X\}$

Input: `retrieve`
call $y \leftarrow$ `retrieve` at int. E of i
output y

Figure A.1: The reduction system C.

The following system equivalences are easy to verify:

$$\mathsf{CT}_1^{\mathsf{H}} \equiv \mathsf{S}^{\mathsf{H}} \tag{A.1}$$

$$\mathsf{CW}_{\mathsf{r}}^{\mathsf{B}} \equiv \mathsf{S}^{\mathsf{B}} \tag{A.2}$$

$$\mathsf{CW}_{\mathsf{i}-1}^{\mathsf{B}} \equiv \mathsf{CW}_{\mathsf{i}}^{\mathsf{H}} \quad \forall i \in \{2, \ldots, q\}. \tag{A.3}$$

As a consequence, we can rewrite the second term as

$$\begin{aligned}
\Delta^{\mathsf{D}''}(\mathsf{S}^{\mathsf{H}}, \mathsf{S}^{\mathsf{B}}) &= \Delta^{\mathsf{D}''}(CT_1^H, CT_r^B) \\
&= \Delta^{\mathsf{D}''}(CT_1^H, CT_1^B) + \Delta^{\mathsf{D}''}(CT_1^B, CT_2^H) \\
&\quad + \Delta^{\mathsf{D}''}(CT_2^H, CT_2^B) + \Delta^{\mathsf{D}}(CT_2^B, CT_3^H) \\
&\quad + \ldots \\
&\quad + \Delta^{\mathsf{D}}(CT_r^H, CT_r^B) \\
&= \sum_{i=1}^{r} \Delta^{\mathsf{D}''}(CT_i^H, CT_i^B) \\
&= r \cdot \Delta^{\mathsf{D}''}(C\mathsf{T}^{\mathsf{H}}, C\mathsf{T}^{\mathsf{B}}) \\
&= r \cdot \Delta^{\mathsf{D}'}(\mathsf{T}^{\mathsf{H}}, \mathsf{T}^{\mathsf{B}}) \\
&= r \cdot \mathbf{Adv}^{\text{RO-CRI}}_{\mathsf{H},\mathsf{f}',\mathsf{R}',\sigma}(\mathsf{D}') + r \cdot \Delta^{\mathsf{D}'}(\mathsf{T}^{\mathsf{RO}}, \mathsf{T}^{\mathsf{B}})
\end{aligned}$$

where in the third step we used Equation (A.3). In the forth step we used that the distinguishing advantage of D'' on the problem with R' is the average of the distinguising advantage of D on resources with the the fixed i. Hence, the sum of these r terms is equal to r times the average.

The overall claim is then directly implied by the following two bounds, which remain to be shown:

$$\Delta^{\mathsf{D}'}(\mathsf{T}^{\mathsf{RO}}, \mathsf{T}^{\mathsf{B}}) \leq \binom{np}{2} 2^{-k'} \tag{A.4}$$

$$\Delta^{\mathsf{D}''}(S^B, S^{RO}) \leq \binom{npr}{2} 2^{-k'} \tag{A.5}$$

In both cases the two resources behave exactly identically until a repeated query to the oracle occurs. Hence, we can bound the distinction advantage by the probability of managing non-adaptively to query twice

the same input [Mau13]. In the following, we only prove A.5, as A.4 follows by an analogous argument.

Let $Z_1, Z_2, \ldots, Z_{npr}$ denote the queries, which are submitted to the beacon. The collision probability can then be bounded using the union bound

$$\Pr(\exists i \neq j \;\; Z_i = Z_j) \leq \sum_{i \neq j} \Pr(Z_i = Z_j).$$

Observe that all queries are either of the form $f(Y_s, A_s)$, where (Y_s, A_s) is the s-th pair output by the entropy source, or $f(Y)$, where Y is an output of the beacon. If either Z_i or Z_j is of the latter type, then the collision probability is trivially upper bounded by $\frac{1}{|H.\mathcal{Y}|} \leq 2^{-k'}$, using that f is injective. If both of them are of the former type, then that by definition of the filter $\mathbf{f}_{p,r}^{\mathsf{r\text{-}splt}}$ the two inputs Z_i, Z_j cannot collide if they depend on the same underlying value X_s from the entropy source. Hence, assume w.l.o.g. that $Y_i = f(Y_s, A_s)$ and $Y_j = f(Y_t, A_t)$ with $s > t$. For every pair of fixed auxiliary information (a_s, a_t), we obtain the following bound:

$$\begin{aligned}
&\Pr(f_i(Y_s, a_s) = f_j(Y_t, a_t)) \\
&\quad = \sum_z \Pr(f_i(Y_s, a_s) = z \wedge f_j(Y_t, a_t) = z) \\
&\quad = \sum_z \Pr(Y_t = f_j^{-1}(z, a_t)) \cdot \Pr(Y_s = f_i^{-1}(z, a_s) \mid Y_t = f_j^{-1}(z, a_t)) \\
&\quad \leq \sum_z \Pr(Y_t = f_j^{-1}(z, a_t)) \cdot \max_{t_s} \Pr(Y_s = t_s \mid Y_t = f_j^{-1}(z, a_t)) \\
&\quad = \sum_{y_t} \Pr(Y_t = y_t) \cdot \max_{y_s} \Pr(Y_s = y_s \mid Y_t = y_t) \\
&\quad = 2^{-\tilde{H}_\infty(Y_s | Y_t)} = 2^{-k} \leq 2^{-k'}.
\end{aligned}$$

Averaging over the choice of (a_s, a_t) yields the desired result

$$\Pr(f_i(Y_s, A_s) = f_j(Y_t, A_t)) \leq 2^{-k'},$$

and, thus, the distinction advantage $\Delta^{\mathsf{D}}(S^B, S^{RO})$ can be bounded as

$$\Pr(\exists i \neq j \;\; Z_i = Z_j) \leq \sum_{i \neq j} \Pr(Z_i = Z_j) \leq \binom{npr}{2} 2^{-k'}. \qquad \square$$

Appendix B

Details of Chapter 5

B.1 Details of Section 5.3

B.1.1 Proof of Theorem 5.3.12

Theorem 5.3.12. *For any resource* R *and any monotone predicates* P_1 *and* P_2*, we have*

$$\begin{aligned}\mathsf{R}^{[P_1,P_2]} &= \bigcup_{n\in\mathbb{N}} \Big(\bigcup \{ \mathsf{R}^{\phi_1\cdot\phi_2\cdots\phi_n} \mid \forall i \leq n : \phi_i \in \{P_2], [P_1\}\} \Big) \\ &= \Big(\big(\mathsf{R}^{[P_1}\big)^{P_2]} \Big)^{[P_1} = \Big(\big(\mathsf{R}^{P_2]}\big)^{[P_1} \Big)^{P_2]},\end{aligned}$$

where $\mathsf{R}^{\phi_1\cdot\phi_2\cdots\phi_n}$ *is a shorthand notation for first applying* ϕ_1*, then* ϕ_2*, until* ϕ_n*.*

Proof. We prove the theorem in three steps:

Claim 1. $\mathcal{R}^{[P_1,P_2]} = \Big(\big(\mathcal{R}^{[P_1}\big)^{P_2]} \Big)^{[P_1}$

Proof of claim. Let $\mathsf{S} \in \Big(\big(\mathcal{R}^{[P_1}\big)^{P_2]} \Big)^{[P_1}$ be arbitrary. This means that

there exist $\mathsf{T} \in \left(\mathcal{R}^{[P_1]}\right)^{P_2]}$, $\mathsf{U} \in \mathcal{R}^{[P_1}$ and $\mathsf{R} \in \mathcal{R}$ such that

$$\text{from}_{P_1}(\mathsf{S}) = \text{from}_{P_1}(\mathsf{T}), \tag{B.1}$$
$$\text{until}_{P_2}(\mathsf{T}) = \text{until}_{P_2}(\mathsf{U}), \tag{B.2}$$
$$\text{from}_{P_1}(\mathsf{U}) = \text{from}_{P_1}(\mathsf{R}). \tag{B.3}$$

Combining these properties with the commutativity of from and until, we obtain

$$\begin{aligned}\text{until}_{P_2}(\text{from}_{P_1}(\mathsf{R})) &= \text{until}_{P_2}(\text{from}_{P_1}(\mathsf{U})) = \text{from}_{P_1}(\text{until}_{P_2}(\mathsf{U})) \\ &= \text{from}_{P_1}(\text{until}_{P_2}(\mathsf{T})) = \text{until}_{P_2}(\text{from}_{P_1}(\mathsf{T})) \\ &= \text{until}_{P_2}(\text{from}_{P_1}(\mathsf{S})),\end{aligned}$$

implying that $\mathsf{S} \in \mathsf{R}^{[P_1,P_2]}$ and thus $\mathcal{R}^{[P_1,P_2]} \supseteq \left(\left(\mathcal{R}^{[P_1]}\right)^{P_2]}\right)^{[P_1}$.

Now, let $\mathsf{S} \in \mathcal{R}^{[P_1,P_2]}$ be arbitrary. By definition, we thus know that there exists a $\mathsf{R} \in \mathcal{R}$ such that $\text{from}_{P_1}(\mathsf{S}) \in \left(\text{from}_{P_1}(\mathsf{R})\right)^{P_2]}$. Combining this with the basic fact that $\text{from}_{P_1}(\mathsf{R}) \in \mathsf{R}^{[P_1}$, we obtain $\text{from}_{P_1}(\mathsf{S}) \in \left(\mathsf{R}^{[P_1}\right)^{P_2]}$. Combining this in turn with the basic fact that $\mathsf{S} \in \mathsf{S}^{[P_1}$ then yields $\mathsf{S} \in \left(\left(\mathsf{R}^{[P_1}\right)^{P_2]}\right)^{[P_1}$ and thus $\mathcal{R}^{[P_1,P_2]} \subseteq \left(\left(\mathcal{R}^{[P_1]}\right)^{P_2]}\right)^{[P_1}$. ◊

Claim 2. $\mathcal{R}^{[P_1,P_2]} = \left(\left(\mathcal{R}^{P_2]}\right)^{[P_1}\right)^{P_2]}$

Proof of claim. Observing that

$$\mathcal{R}^{[P_1,P_2]} = \left\{\mathsf{S} \;\middle|\; \text{until}_{P_2}(\text{from}_{P_1}(\mathsf{R})) = \text{until}_{P_2}(\text{from}_{P_1}(\mathsf{S}))\right\}$$

(using the commutativity), it is easy to see that the proof follows analogously. ◊

Claim 3. $\mathcal{R}^{[P_1,P_2]} = \bigcup_{n\in\mathbb{N}}\left(\bigcup\left\{\mathcal{R}^{\phi_1\cdot\phi_2\cdots\phi_n} \mid \forall i \leq n : \phi_i \in \{P_2],[P_1\}\right\}\right)$

Proof of claim. Using the previous claims, it is trivial to see that

$$\mathcal{R}^{[P_1,P_2]} \subseteq \bigcup_{n\in\mathbb{N}}\left(\bigcup\left\{\mathcal{R}^{\phi_1\cdot\phi_2\cdots\phi_n} \mid \forall i \leq n : \phi_i \in \{P_2],[P_1\}\right\}\right).$$

For the other direction, we proceed by induction over n. Note that without loss of generality (by Theorems 5.3.3 and 5.3.8), we can assume the order of the relaxations to strictly alternate. Furthermore, the previous claims already prove it for $n = 3$. Hence, the relation is also trivial for $n < 3$, as adding a further relaxation only enlarges the set. Assume now as the induction hypothesis that for some $n \geq 3$

$$\mathcal{R}^{[P_1,P_2]} \supseteq \bigcup \{ \mathcal{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \mid \forall i \leq n : \phi_i \in \{P_2], [P_1\} \}.$$

We want to show that $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_{n+1}} \in \mathsf{R}^{[P_1,P_2]}$ as well. Assume w.l.o.g. that $\phi_{n+1} = P_2]$. By the induction hypothesis, we have that $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \in \mathsf{R}^{[P_1,P_2]}$ and thus by the second claim $\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \in \left(\left(\mathsf{R}^{P_2]} \right)^{[P_1} \right)^{P_2]}$ and thus

$$\mathsf{R}^{\phi_1 \cdot \phi_2 \cdots \phi_n} \in \left(\left(\left(\mathsf{R}^{P_2]} \right)^{[P_1} \right)^{P_2]} \right)^{P_2]} = \left(\left(\left(\mathsf{R}^{P_2]} \right)^{[P_1} \right)^{P_2]} \right)^{P_2]},$$

where the second step follows from Theorem 5.3.3. ◊

B.1.2 Proof of Theorem 5.3.16

Theorem 5.3.16. *Let P_1 and P_2 be two monotone predicates, and let ϵ be a function mapping distinguishers to values in $[0, 1]$. Then, for any specification $\mathcal{R}$ we have*

$$\left(\mathcal{R}^{[P_1,P_2]:\epsilon} \right)^{[P_1',P_2']:\epsilon'} \subseteq \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']:\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']} + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}},$$

where $\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']}(D) \coloneqq \epsilon(\mathsf{D} \circ \mathrm{until}_{P_2 \vee P_2'} \circ \mathrm{from}_{P_1 \wedge P_1'})$, i.e., the performance of the distinguisher interacting with the projected resource, and analogously for $\epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}$.

Proof. Observe that it suffices to show

$$\left(\left(\left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon} \right)^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon'} \subseteq \left(\left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon_{[P_1,P_2]} + \epsilon'_{[P_1,P_2]}} \right)^{[P_1 \wedge P_1', P_2 \vee P_2']}.$$

The rest then follows trivially by Theorems 2.2.10 and 5.3.13. To this end, consider an arbitrary $\mathsf{S} \in \left(\left(\left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon} \right)^{[P_1 \wedge P_1', P_2 \vee P_2']} \right)^{\epsilon'}$.

Hence, there must exist a $\mathsf{T} \in \left(\left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}\right)^{\epsilon}\right)^{[P_1 \wedge P_1', P_2 \vee P_2']}$, a $\mathsf{U} \in \left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}\right)^{\epsilon}$, and a $\mathsf{V} \in \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}$ such that

$$\left|\Delta^{\mathsf{D}}(\mathsf{S}, \mathsf{T})\right| \leq \epsilon'(\mathsf{D}) \tag{B.4}$$

$$\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{T})) = \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{U})) \tag{B.5}$$

$$\left|\Delta^{\mathsf{D}}(\mathsf{U}, \mathsf{V})\right| \leq \epsilon(\mathsf{D}). \tag{B.6}$$

Using those properties, we obtain

$$\begin{aligned}
&\left|\Delta^{\mathsf{D}}\left(\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{V})), \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{S}))\right)\right| \\
&\leq \left|\Delta^{\mathsf{D}}(\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{V})), \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{U})))\right| \\
&\qquad + \left|\Delta^{\mathsf{D}}(\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{T})), \mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{S})))\right| \\
&\leq \left|\Delta^{\mathsf{D} \circ \mathrm{until}_{P_2 \vee P_2'}(\cdot) \circ \mathrm{from}_{P_1 \wedge P_1'}(\cdot)}(\mathsf{V}, \mathsf{U})\right| \\
&\qquad + \left|\Delta^{\mathsf{D} \circ \mathrm{until}_{P_2 \vee P_2'}(\cdot) \circ \mathrm{from}_{P_1 \wedge P_1'}(\cdot)}(\mathsf{T}, \mathsf{S})\right| \\
&\leq \epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']}(\mathsf{D}) + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}(\mathsf{D}).
\end{aligned}$$

Now observe that $\mathsf{V} \in \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}$ implies

$$\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{V})) \in \mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']},$$

and thus,

$$\mathrm{until}_{P_2 \vee P_2'}(\mathrm{from}_{P_1 \wedge P_1'}(\mathsf{S})) \in \left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}\right)^{\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']} + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}}.$$

As a result, we have

$$\mathsf{S} \in \left(\left(\mathcal{R}^{[P_1 \wedge P_1', P_2 \vee P_2']}\right)^{\epsilon_{[P_1 \wedge P_1', P_2 \vee P_2']} + \epsilon'_{[P_1 \wedge P_1', P_2 \vee P_2']}}\right)^{[P_1 \wedge P_1', P_2 \vee P_2']},$$

concluding the proof. □

B.1.3 Proof of Proposition 5.3.20

Proposition 5.3.20. *Let $\boldsymbol{\pi}_{\mathsf{ENC}} = (\pi_{\mathsf{enc}}, \pi_{\mathsf{dec}})$ denote the protocol securing communication using a symmetric encryption scheme. Then, for*

the resources in Figure 5.1, there exist (efficient) simulators $\boldsymbol{\sigma_1}$*,* $\boldsymbol{\sigma_2}$*, and* $\boldsymbol{\sigma_3}$ *such that*

$$\bigwedge_{(\boldsymbol{\sigma},\epsilon,P_1,P_2)\in\Omega} \Big(\big[\mathsf{EncKey}, \mathsf{AuthChDg}\big] \xrightarrow[\mathsf{interval}]{\boldsymbol{\pi}_{\mathsf{ENC}},\boldsymbol{\sigma},\epsilon,[P_1,P_2]} \mathsf{SecChDg} \Big)$$

for

$$\Omega \coloneqq \big\{ \big(\boldsymbol{\sigma}_1, \epsilon_{\mathsf{CPA}}, \mathtt{true}, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}\big), \big(\boldsymbol{\sigma}_2, 0, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}, \mathtt{false}\big), \big(\boldsymbol{\sigma}_3, 0, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}, \mathtt{false}\big) \big\},$$

where ϵ_{CPA} *denotes a simple reduction from distinguishing the secure and authenticated channel (without key leakage) to the IND-CPA game.*

Proof (Sketch). Let $\boldsymbol{\sigma}_1$ be the usual simulator that creates fake ciphertexts by encrypting $0^{|m|}$ instead of m, when queried for the leakage of the channel. By definition, we have $\mathsf{until}_{\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}}(\boldsymbol{\sigma}_1\mathsf{SecChDg}) \in (\boldsymbol{\sigma}_1\mathsf{SecChDg})^{[\mathtt{true}, \mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}]}$. Thus, if the encryption scheme is IND-CPA secure, then it is easy to see that

$$\mathsf{until}_{\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}}(\boldsymbol{\pi}_{\mathsf{ENC}}\big[\mathsf{EncKey}, \mathsf{AuthChDg}\big]) \in \big((\boldsymbol{\sigma}_1\mathsf{SecChDg})^{[\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}} \vee \mathcal{E}^{\mathsf{leaked}}_{\mathsf{AuthKey}}]}\big)^{\epsilon_{\mathsf{CPA}}},$$

which proves the construction inside the first interval. For the second interval, consider the simulator $\boldsymbol{\sigma}_2$ that encrypts the real messages. Using correctness of the encryption scheme, it is easy to see that

$$\mathsf{from}_{\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}}(\boldsymbol{\pi}_{\mathsf{ENC}}\big[\mathsf{EncKey}, \mathsf{AuthChDg}\big]) = \mathsf{from}_{\mathcal{E}^{\mathsf{leaked}}_{\mathsf{EncKey}}}(\boldsymbol{\sigma}_2\mathsf{SecChDg}),$$

and analogous for the third interval using the same simulator $\boldsymbol{\sigma}_3 = \boldsymbol{\sigma}_2$. □

B.2 Details of Section 5.4

B.2.1 ElGamal Commitments

In this section, we provide some additional details to the ElGamal-commitment example. First, a formal definition of the corresponding

converter implementing the protocol is presented in Figure B.1. Second, a formal definition of the two simulators involved in the construction statements of is depicted in Figure B.2. The left simulator $\sigma^{\mathsf{A}}_{\mathsf{ElG\text{-}com}}$ is used to formalize security against a potentially dishonest Alice (initiator), and the right $\sigma^{\mathsf{B}}_{\mathsf{ElG\text{-}com}}$ is used to formalize security against a potentially dishonest Bob (responder), respectively.

B.2.2 Coin-Tossing

In this section, we provide some additional details on the coin-tossing example. First, in Figure B.3 we prove the formal definition of the two converters of Blum's protocol for constructing a single-bit coin-toss resource. Note that in the protocol Alice acts as the initiator, and Bob as the responder, respectively. The pseudo-code description is presented in Figure B.3.

Second, a formal description of the two simulators involved in the construction statements of the coin-toss resource is depicted in Figure B.4. The left $\sigma^{\mathsf{A}}_{\mathsf{CT}}$ is used to formalize security against a potentially dishonest Alice (initiator), and the right $\sigma^{\mathsf{B}}_{\mathsf{CT}}$ is used to formalize security against a potentially dishonest Bob (responder), respectively.

Converter $\pi^{\mathsf{A}}_{\mathsf{EIG\text{-}com}}$

Initialization

$a, b, v \leftarrow \perp$

Emulating Interface A of $\mathsf{Com}^{\mathsf{A}\to\mathsf{B}}_{\mathbb{G}}$

Input: $(\mathtt{commit}, m)$
require only called once
$a, b \twoheadleftarrow \mathbb{Z}_{|\mathbb{G}|}$
$v \leftarrow m$
call $(\mathtt{send}, (g^a, g^b, m \cdot g^{ab}))$
at interface A of $\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{committed}}$
return ok

Input: $(\mathtt{open}, m)$
require $\mathcal{E}^{\mathsf{committed}}$ and only called once
if $m = v$ **then**
call $(\mathtt{send}, (a, b))$
at interface A of $\mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{opened}}$
return ok

Converter $\pi^{\mathsf{B}}_{\mathsf{EIG\text{-}com}}$

Initialization

$c, A, B, v \leftarrow \perp$

Emulating Interface B of $\mathsf{Com}^{\mathsf{A}\to\mathsf{B}}_{\mathbb{G}}$

Input: isCommitted
if $c = \perp$ **then**
call $(c, A, B) \leftarrow \mathtt{receive}$
at interface B of $\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}$
return $c \neq \perp$

Input: read
if $c = \perp$ **then**
call $(c, A, B) \leftarrow \mathtt{receive}$
at interface B of $\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}$
if $v = \perp$ **then**
call $(a, b) \leftarrow \mathtt{receive}$
at interface B of $\mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}$
if $(a, b) \neq \perp \wedge A = g^a \wedge B = g^b$ **then**
$v \leftarrow c \cdot g^{-ab}$
return v

Figure B.1: A description of the ElGamal-commitment converters.

Converter $\sigma^{\mathsf{B}}_{\mathsf{EIG\text{-}com}}$

Initialization

$u, v, w \twoheadleftarrow \mathbb{G}$
$x \leftarrow \perp$

Emulating Interface B of $\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}$

Input: receive
call $succ \leftarrow \mathtt{isCommitted}$
at interface B of $\mathsf{Com}^{\mathsf{A}\to\mathsf{B}}_{\mathbb{G}}$
if $succ$ **then**
return (u, v, w)
else
return $\perp$

Emulating Interface B of $\mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}$

Input: receive
return $\perp$

Converter $\sigma^{\mathsf{A}}_{\mathsf{EIG\text{-}com}}$

Initialization

$A, B, V \leftarrow \perp$

Emulating Interface A of $\mathsf{Ch}_1^{\mathsf{A}\to\mathsf{B}}$

Input: $(\mathtt{send}, x)$
require only called once
if $x \in \mathbb{G}^3$ **then** $(A, B, V) \leftarrow x$
call $(\mathtt{commitRaw}, x)$
at interface A of $\mathsf{Com}^{\mathsf{A}\to\mathsf{B}}_{\mathbb{G}}$
return ok

Emulating Interface A of $\mathsf{Ch}_2^{\mathsf{A}\to\mathsf{B}}$

Input: $(\mathtt{send}, x)$
require only called once
if $x{=}(a, b) \in \mathbb{Z}_{|\mathbb{G}|} \wedge A{=}g^a \wedge B{=}g^b$ **then**
call $(\mathtt{open}, V \cdot g^{-ab})$
at interface A of $\mathsf{Com}^{\mathsf{A}\to\mathsf{B}}_{\mathbb{G}}$
return ok

Figure B.2: The simulators from the ElGamal-commitment construction.

Converter $\pi^{\mathsf{A}}_{\mathsf{CT}}$

Initialization

$X, Y \leftarrow \bot$

Emulating Interface A of $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\}}$

Input: `initiate`
- **require** only called once
- $X \twoheadleftarrow \{0,1\}$
- **call** `commit` at int. A of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}$
- **return** `ok`

Input: `read`
- **require** $X \neq \bot$
- **if** $Y = \bot$ **then**
  - **call** $Y \leftarrow$ `read` at int. A of $\mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}$
- **if** $Y \neq \bot$ **then return** $X \oplus Y$
- **else return** $\bot$

Input: `release`
- **require** $Y \neq \bot$ and only called once
- **call** $(\mathtt{open}, X)$ at int. A of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}$
- **return** `ok`

Converter $\pi^{\mathsf{B}}_{\mathsf{CT}}$

Initialization

$X, Y \leftarrow \bot$

Emulating Interface B of $\mathsf{CT}^{\mathtt{A},\mathtt{B}}_{\{0,1\}}$

Input: `isInitiated`
- **call** $b \leftarrow$ `isCommitted` at interface B of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}$
- **return** b

Input: `respond`
- **require** `isInitiated` and only called once
- $Y \twoheadleftarrow \{0,1\}$
- **call** $(\mathtt{send}, Y)$ at int. B of $\mathsf{Ch}^{\mathtt{B}\to\mathtt{A}}$
- **return** `ok`

Input: `read`
- **require** $Y \neq \bot$
- **if** $X = \bot$ **then**
  - **call** $X \leftarrow$ `read` at interface B of $\mathsf{Com}^{\mathtt{A}\to\mathtt{B}}$
- **if** $X \neq \bot$ **then**
  - **return** $X \oplus Y$
- **else**
  - **return** $\bot$

Figure B.3: A formal description of the coin-tossing converters.

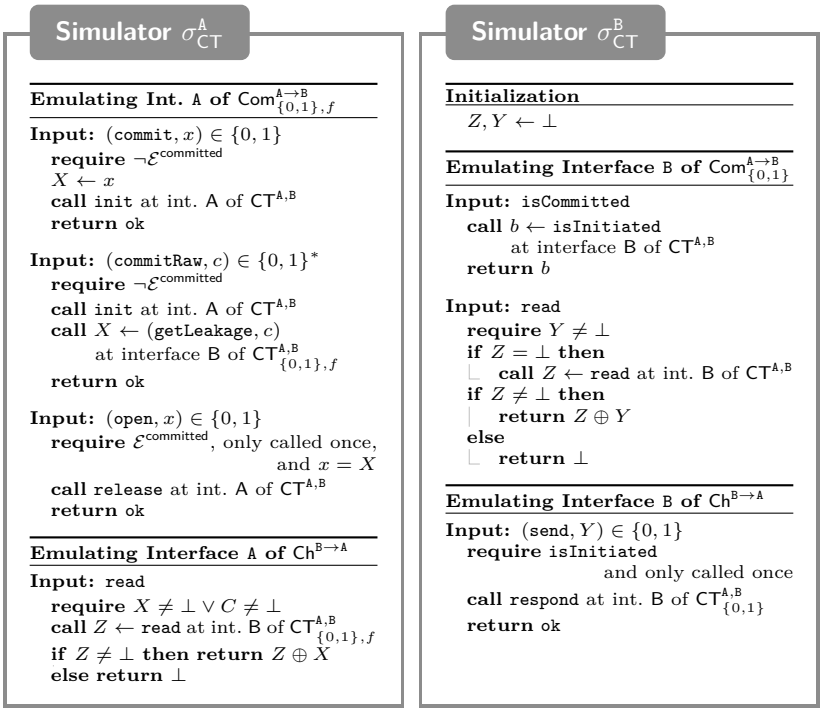


Figure B.4: A description of the respective coin-tossing simulators.

Appendix C

Details of Chapter 6

C.1 Details of Section 6.3.1

C.1.1 Key-Updating Signatures

Syntax. A key-updating signature scheme KuSig consists of three polynomial-time algorithms ($\mathsf{KuSig.Gen}, \mathsf{KuSig.Sign}, \mathsf{KuSig.Verify}$). The probabilistic algorithm $\mathsf{KuSig.Gen}$ generates an initial signing key sk and a corresponding verification key vk. Given a message m and sk, the signing algorithm outputs an updated signing key and a signature: $(sk', \sigma) \leftarrow \mathsf{KuSig.Sign}(sk, m)$. Similarly, the verification algorithm outputs an updated verification key and the result v of verification: $(vk', v) \leftarrow \mathsf{KuSig.Verify}(vk, m, \sigma)$.

Correctness. Let (sk_0, vk_0) be any output of $\mathsf{KuSig.Gen}$, and let $m_1, \ldots, m_k$ be any sequence of messages. Further, let $(sk_i, \sigma_i) \leftarrow \mathsf{KuSig.Sign}(sk_{i-1}, m_i)$ and $(vk_i, v_i) \leftarrow \mathsf{KuSig.Verify}(vk_{i-1}, m_i, \sigma_i)$ for $i = 1 \ldots (k-1)$. For correctness, we require that $v_i = 1$ for all $i = 1 \ldots (k-1)$.

Security. The security of KuSig is formalized using the game KuSig-UF, described in Figure C.1. For simplicity, we define the security in the single-user setting (security in the multi-user setting can be obtained using the standard hybrid argument).

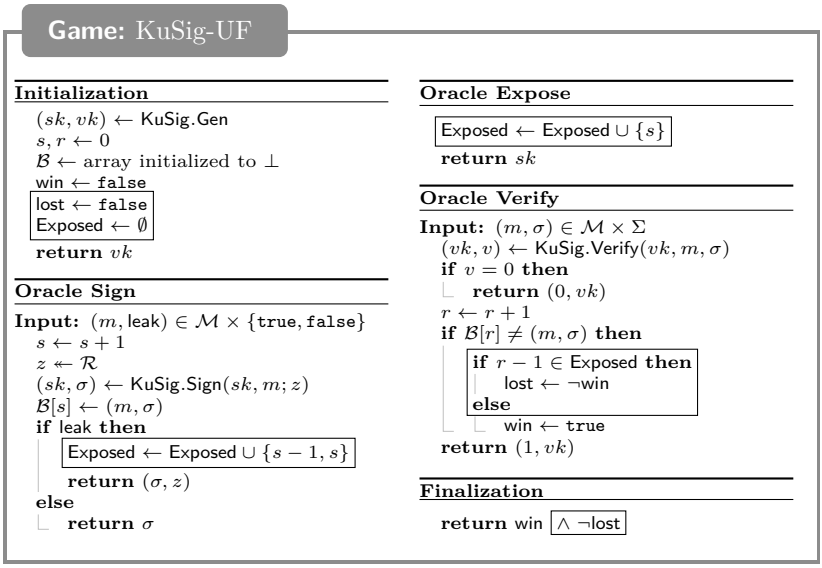


Figure C.1: The strong unforgeability game for key-updating signatures.

C.1.2 The Authentication Protocol

Recall that in protocol, whenever the sender wants to send a message, a fresh signing and verification key pair is sampled. The fresh verification key is then signed together with the message—using the prior signing key— and the message, the verification key and the signature are transmitted. Finally, the old signing key is securely erased and the fresh one stored instead. Moreover, with each message, the sender also transmits a hash of the previous verification key. The receiver, on the other hand verifies a received message with the previous verification key and stores the new one. See Figure C.2 for a formal definition of the two converters.

C.1.3 Proof of Theorem 6.3.1

Proof. See Figure C.3 for a description of the simulator. Note that the simulator is parameterized in the real world's security guarantees, e.g.,

Converter sig_i

Emulating Interface A of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$

Input: $(\mathtt{send}, m, ad) \in \mathcal{M} \times \mathcal{AD}$
require only once & isAvailable

// Generate fresh keys
call $r \leftarrow$ `sample` at int. A of $\mathsf{Rand}^{kg_i,\mathtt{A}}$
$(sk, vk) \leftarrow \mathsf{Sig.Gen}(r)$

// Send the verif. key along the msg
call $(\mathtt{write}, m, (ad, vk))$
at interface A of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$

// Store the keys
call $(\mathtt{write}, (sk, vk))$
at interface A of $\mathsf{Mem}^{sk_i,\mathtt{A}}$
return `ok`

Input: `isAvailable`
call $succ \leftarrow$ `isAvailable`
at interface A of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
return $succ$

Emulating Interface A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$

Input: $(\mathtt{send}, m, ad) \in \mathcal{M} \times \mathcal{AD}$
require only once & isAvailable

// Fetch the signing key
call $(sk, vk) \leftarrow$ `read`
at interface A of $\mathsf{Mem}^{sk,\mathtt{A}}$

// Sign and send the message
$\sigma \leftarrow \mathsf{Sig.Sign}(sk, (m, ad))$
$h \leftarrow \mathrm{hash}(ad, vk)$
call $(\mathtt{write}, (m, h, \sigma), ad)$
at interface A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$

// Erase the signing key
call `erase` at int. A of $\mathsf{Mem}^{sk_i,\mathtt{A}}$
return `ok`

Input: `isAvailable`
call $succ \leftarrow$ `isAvailable`
at interface A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
call $(sk, vk) \leftarrow$ `read`
at interface A of $\mathsf{Mem}^{sk_i,\mathtt{A}}$
return $succ \wedge ((sk, vk) \neq \bot)$

Converter vrf_i

Emulating Interface B of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$

Input: `receive`
require only called once

// Receive the message with the vk
call $(m, (ad, vk)) \leftarrow$ `read`
at interface B of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
if $(m, (ad, vk)) = \bot$ **then return** $\bot$

// Store the verification key
call `write`, vk at int. B of $\mathsf{Mem}^{vk_i,\mathtt{B}}$
return (m, ad)

Input: `isAvailable`
call $succ \leftarrow$ `isAvailable`
at interface B of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
return $succ$

Emulating Interface B of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$

Input: `receive`
require only once & isAvailable

// Receive the message
call $((m, h, \sigma), ad) \leftarrow$ `read`
at interface B of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
if $((m, h, \sigma), ad) = \bot$ **then return** $\bot$

// Verify the signature and hash
call $vk \leftarrow$ `read` at int. B of $\mathsf{Mem}^{vk_i,\mathtt{B}}$
$v \leftarrow \mathsf{Sig.Verify}(vk, (m, ad), \sigma)$
$h' \leftarrow \mathrm{hash}(vk, ad)$
if $\neg v \vee h \neq h'$ **then return** $\bot$
return (m, ad)

Input: `isAvailable`
call $succ \leftarrow$ `isAvailable`
at interface B of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
call $vk \leftarrow$ `read` at int. A of $\mathsf{Mem}^{vk_i,\mathtt{B}}$
return $succ \wedge (vk \neq \bot)$

Figure C.2: A description of the converters sig_i and vrf_i implementing the unidirectional authentication scheme for a single message.

the can-leak predicate of the memory resources. We now proceed to argue that this simulator actually makes the two worlds indistinguishable.

The simulator internally samples a signing-verification key pair and remembers the randomness. Using this, it is easy see to that Alice's memory and randomness resources can be perfectly simulated, since the simulator knows when Alice sent her messages from the corresponding events and the real-world can-leak predicates. Moreover, Bob's memory storing the verification key he receives is also simple to emulate, since the key is transmitted as part of the associated data the simulator sees.

Next, consider the $(i-1)$-st channel. The can-send predicate of the ideal world enforces that the sent event is triggered if and only if it is triggered in the real world. The leakage at Eve's interface is then also simple to simulate: the simulator just appends the verification key to the associated data. Handling injections is also straight-forward: in the real-world, if the distinguisher asks for $(m', (ad', vk'))$ to be injected, then at the point of time Bob fetches this will differ from $(m, (ad, vk))$ if and only one of the components differ and the corresponding injection-function will be evaluated. In the ideal world, the simulator removes vk' and asks to inject (m', ad'), setting the *same* flag to `false` if $vk' \neq vk$, leading to the same behavior. If the distinguisher request a specific error to be triggered, the simulator can simply forward this as well, thereby excluding the signature-verification error.

The interesting part to simulate is everything with respect to the i-th channel. First, observe that Alice will only send the message after sending the $(i-1)$-st, which the can-send predicate in the ideal world ensures as well. Simulating the leakage is also straightforward: the simulator just adds the hash and the signature himself. Now consider a delivery attempt. First, the simulator only processes it once there has also been a delivery on the former channel, such that it knows the verification key Bob will use. Since Bob will not fetch the latter message out-of-order, as enforced by the protocol and the can-receive predicates, respectively, this is not observable, however. Recall that

$$\mathfrak{D}^{\mathsf{S}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}, same) := \begin{cases} err & \text{if } \mathfrak{D}^{\mathsf{R}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\tilde{\mathcal{E}}, same) = err \\ & \quad \wedge err \neq \mathtt{msg} \\ \mathsf{sig\text{-}err} & \text{else if } \neg(same \vee \mathcal{E}_i^{\mathsf{sk\text{-}known}}) \\ \mathtt{msg} & \text{else} \end{cases}$$

Thus, to show that the two worlds behave identically, we need (1) show that the *same* flag is consistent, as otherwise the channels might trigger different error before it even gets to the signature verification, (2) show that the two worlds trigger a signature-verification error for the same inputs. We consider two cases:

An explicit delivery ($(m', h', \sigma') \neq \texttt{fwd}$):

1. In order for the ideal-world to behave equivalent, the simulator needs to ensure that the *same* flag in both worlds agree. In the real-world, however, this flag takes the signature and the hash into account, which in the ideal-world are not part of the message but only simulated. The simulator, thus, needs to detect any modification of the signature and the hash itself, and if necessary enforce $same = \mathsf{false}$. Note that the simulator might never see Alice's message, since the channel might be confidential, and thus cannot trivially check this. It thus proceeds as follows: it verifies the signature using the original verification key (not necessarily the one used by Bob). By correctness of the signature scheme, a failure clearly indicates that something must have been tampered with. By uniqueness of the signatures, it moreover follows that if $(m', ad') = (m, ad)$ and the verification succeeds, then it must have been the same signature. Hence, the simulator can enforce $same = \texttt{false}$ whenever the verification fails, resulting in the channel using $same = \texttt{false}$ iff $(m, \sigma, ad) \neq (m', \sigma', ad')$. For the hash value, the simulator can recompute $\mathrm{hash}(ad', vk)$, and set $same = \mathsf{false}$ whenever it does not match. If $ad = ad'$, this check ensures that $h' = h$, and otherwise the channel will set $same = \mathsf{false}$ anyways. Thus, we know that in the real-world an error happens before the signature verification iff the same happens in the idea-world.
2. Now consider the signature verification and the hash check. For the former, the simulator simply verifies the signature using vk' (which Bob uses in the real-world). If the check succeeds, it injects the message. Otherwise, it triggers a signature-failure event. It remains to see that whenever the simulator tries to inject the message this is actually allowed by $\mathfrak{D}^{\mathsf{S}_i^{\mathsf{auth}}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}, same)$. It is, however, easy to see that this happening would directly

imply an existential forgery, since it means that Eve injected a different message with a valid signature, with respect to the correct verification key, and without having any information about the signing key (neither the key nor its randomness leaked). The simulator can furthermore easily check that $h' = \text{hash}(ad', vk')$, this time using vk', which also Bob would use in the real-world.

A forwarding request: Whenever the distinguisher asks to deliver the original message, but with a potentially different associated data, the *same* flag is trivially to decide: its the same iff the associated data match. Since this is what the channel checks anyways, there is nothing for the simulator to be taken care of. Now consider the check performed by Bob's protocol. In the real world, by collision resistance, the hashes will match iff $ad = ad'$ and $vk = vk'$. In this case, by correctness, the signature verification will also succeed. Hence, the protocol accepts iff $ad = ad'$ and $vk = vk'$, which can easily be emulated.

If the distinguisher asks to trigger an error, this can be handled analogous to the delivery request. More precisely, the simulator decides *same* analogously and then requests an the error to be triggered, thereby excluding the signature error. □

Simulator σ_{auth}

Let ℓ_{sig} and ℓ_{hash} the length of a signature and a hash, respectively.

Initialization

$r \twoheadleftarrow \mathcal{R}$
$(sk, vk) \leftarrow \mathsf{Sig.Gen}(r)$
$vk' \leftarrow \bot$

Emulating Interface E of $\mathsf{Mem}^{sk_i,\mathtt{A}}$

Input: `read`
if $\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i-1,\mathtt{A}\to\mathtt{B})} \wedge \neg\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \wedge \mathfrak{L}^{\mathsf{R}^{\mathsf{auth}}_i}_{\mathsf{Mem}(sk_i,\mathtt{A})}(\tilde{\mathcal{E}})$ **then**
  $\mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(sk_i,\mathtt{A})}$
  return (sk, vk)
else
  return $\bot$

Emulating Interface E of $\mathsf{IMem}^{vk_i,\mathtt{B}}$

Input: `read`
if $\mathcal{E}^{\mathsf{received}}_{\mathsf{Ch}(i-1,\mathtt{A}\to\mathtt{B})}$ **then return** vk'
else return $\bot$

Emulating Interface E of $\mathsf{Rand}^{kg_i,\mathtt{A}}$

Input: `triggerLeaking`
if $\neg\mathfrak{L}^{\mathsf{R}^{\mathsf{auth}}}_{\mathsf{Rand}(kg_i,\mathtt{A})}(\tilde{\mathcal{E}})$ **then return** $\bot$
$\mathcal{E} \overset{+}{\leftarrow} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(kg_i,\mathtt{A})}$
return ok

Input: `getLeakage`
if $\neg\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \vee \neg\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(kg_i,\mathtt{A})}$ **then**
  return $\bot$
return r

Emulating Interface E of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$

Input: `read`
  call $(m, ad) \leftarrow$ `read`
    at interface E of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
  if $(m, ad) = \bot$ **then return** $\bot$
  else return $(m, (ad, vk))$

Input: `readLength`
  call $(\ell, ad) \leftarrow$ `readLength`
    at interface E of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
  if $(\ell, ad) = \bot$ **then return** $\bot$
  else return $(\ell, (ad, vk))$

Input: $(\mathtt{deliver}, m', (ad', vk''), same)$
  $vk' \leftarrow vk''$
  if $same = \mathtt{check} \wedge vk'' \neq vk$ **then**
    $same \leftarrow \mathtt{false}$
  call $(\mathtt{deliver}, m', ad', same)$
    at interface E of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
  return `ok`

Input: $(\mathtt{error}, err, m', (ad', vk''), Overw, same)$
  if $same = \mathtt{check} \wedge vk'' \neq vk$ **then**
    $same \leftarrow \mathtt{false}$
  $O' \leftarrow Overw \cup \{\mathsf{sig\text{-}err}\}$
  call $(\mathtt{error}, err, O', m', ad', same)$
    at interface E of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
  return `ok`

Emulating Interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$

Input: $(\mathtt{error}, err, Overw, (m', h', \sigma'), ad', same)$
  if $same = \mathtt{check}$ **then**
    handle as in deliver by computing v_{S} and h_{S}
  $Overw \leftarrow Overw \cup \{\mathsf{sig\text{-}err}\}$
  call $(\mathtt{error}, err, Overw, m', ad', same)$
    at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  return `ok`

Input: `read`
  call $(m, ad) \leftarrow (\mathtt{read}, m, ad)$
    at interface E of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
  if $(m, ad) = \bot$ **then return** $\bot$
  $\sigma \leftarrow \mathsf{Sig.Sign}(sk, (m, ad))$
  return $((m, h(vk, ad), \sigma), ad)$

Emulating Int. E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$ (cont.)

Input: `readLength`
  call $(\ell, ad) \leftarrow$ `readLength`
    at interface E of $\mathsf{Ch}^{i-1,\mathtt{A}\to\mathtt{B}}$
  if $(\ell, ad) = \bot$ **then return** $\bot$
  return $(\ell + \ell_{\mathsf{sig}} + \ell_{\mathsf{hash}}, ad)$

Input: $(\mathtt{deliver}, (m', h', \sigma'), ad', same)$
  if $(m', h', \sigma'), = \mathtt{fwd} \wedge \neg \mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}}$ **then**
    return $\bot$

At this point, if delivery of $(i-1)$-st channel not called, then output `ok` immediately but delay processing until then.

  if $(m', h', \sigma') = \mathtt{fwd}$ **then**
    call $(\ell, ad) \leftarrow$ `readLength`
      at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
    if $ad' = ad \wedge vk' = vk$ **then**
      call $(\mathtt{deliver}, \mathtt{fwd}, ad', same)$
        at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
    else
      if $same = \mathtt{check}$ **then**
        $same \leftarrow ad' = ad$
      call $(\mathtt{error}, \mathsf{sig\text{-}err}, \emptyset, \bot, \bot, same)$
        at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  else
    $v_{\mathsf{S}} \leftarrow \mathsf{Sig.Verify}(vk, (m', ad'), \sigma)$
    $v_{\mathsf{R}} \leftarrow \mathsf{Sig.Verify}(vk', (m', ad'), \sigma')$
    $h_{\mathsf{S}} \leftarrow \mathrm{hash}(ad, vk)$
    $h_{\mathsf{R}} \leftarrow \mathrm{hash}(ad', vk)$
    if $same = \mathtt{check}$ **then**
      if $\neg v_{\mathsf{S}} \vee h_{\mathsf{S}} \neq h'$ **then**
        $same \leftarrow \mathtt{false}$
    if $v_{\mathsf{R}} \wedge h_{\mathsf{R}} = h'$ **then**
      call $(\mathtt{deliver}, m', ad', same)$
        at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
    else
      call $(\mathtt{error}, \mathsf{sig\text{-}err}, \emptyset, m', ad', same)$
        at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  return `ok`

Figure C.3: The simulator for Theorem 6.3.1.

C.2 Details of Section 6.3.2

C.2.1 The Sesqui-directional HIBE Protocol

Recall that the protocol proceeds in epochs, where each epoch is initiated by Bob sending a fresh HIBE public key mpk (and indicating how many messages he received at this moment). Within the epoch, Alice sends then a sequence of messages to Bob, encrypted under this public key and using as identity (the hashes of) all ciphertexts she sent since the message indicated by Bob. See Figure C.4 for a schematic depiction of the scheme.

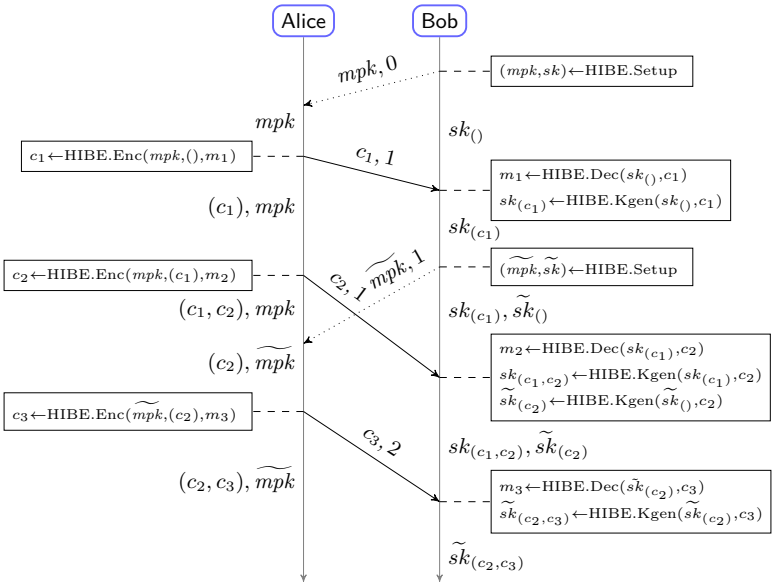


Figure C.4: Sesqui-directional confidentiality from HIBE, depicting the first and the beginning of the second epoch.

See Figures C.5 and C.6 for a formal definition of the two converters hibe-enc and hibe-dec, respectively, that implement this protocol when connected to the real-world resource $\mathsf{R}^{\mathsf{hibe}}$. Note that in the formal definition we allow ourselves to be a bit sloppy when it comes to determine

in which epoch a party is, or how many messages a party already sent or received. While this in principle can be decuded from the memory resources the parties have available, a reasonable implementation would of course just store this (public) information directly.

Converter hibe-enc

Emulating Int. A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}, i \in [n]$

Input: $(\mathtt{send}, m, ad) \in \mathcal{M} \times \mathcal{AD}$
  require only called once & isAvailable

  // Fetch the mpk
  call $(j, mpk, rcv) \leftarrow \mathtt{read}$
    at interface A of $\mathsf{IMem}^{pk,\mathtt{A}}$

  // Compute the identity
  $tr \leftarrow ()$
  for $k = (rcv + 1), \ldots, i - 1$ **do**
    call $h_k \leftarrow \mathtt{read}$
      at interface A of $\mathsf{Mem}^{tr_k,\mathtt{A}}$
    $tr \leftarrow tr \parallel h_k$

  // Encrypt and send
  call $r \leftarrow \mathtt{sample}$ at int. A of $\mathsf{Rand}^{enc_i,\mathtt{A}}$
  $c \leftarrow \mathrm{HIBE.Enc}(mpk, tr, m; r)$
  call $succ \leftarrow (\mathtt{write}, c, (j, ad))$
    at interface A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$

  // Store the hash
  call $(\mathtt{write}, \mathrm{hash}(c, j, ad))$
    at interface A of $\mathsf{Mem}^{tr_i,\mathtt{A}}$
  return ok

Input: isAvailable
  call $succ \leftarrow \mathtt{isAvailable}$
    at interface A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  call $(j, mpk, rcv) \leftarrow \mathtt{read}$
    at interface A of $\mathsf{IMem}^{pk,\mathtt{A}}$
  $succ \leftarrow succ \wedge ((j, mpk, rcv) \neq \bot)$
  if $i > 1$ **then**
    call $tr \leftarrow \mathtt{read}$
      at interface A of $\mathsf{Mem}^{tr_{i-1},\mathtt{A}}$
    $succ \leftarrow succ \wedge (tr \neq \bot)$
  return $succ$

Emulating Int. A of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}, j \in [n]$

Input: receive
  require only called once

  // Read the message
  call $(m, (mpk, rcv, ad)) \leftarrow \mathtt{read}$
    at interface A of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  if $(m, (mpk, r, ad)) = \bot$ **then**
    return $\bot$

  // Store the mpk
  call $(\mathtt{write}, (j, mpk, rcv))$
    at interface A of $\mathsf{IMem}^{pk,\mathtt{A}}$
  return (m, ad)

Input: isAvailable
  call $succ \leftarrow \mathtt{isAvailable}$
    at interface A of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  call $(j', mpk, rcv) \leftarrow \mathtt{receive}$
    at interface A of $\mathsf{IMem}^{pk,\mathtt{A}}$
  return $succ \wedge (j', mpk, rcv) \neq \bot \wedge j = j' + 1$

Figure C.5: A formal description of the converter hibe-enc for the sender.

Converter hibe-dec

Emulating Int. B **of** $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}, i \in [n]$

Input: `receive`
  require only called once & isAvailable
  call $(c, (ep, ad)) \leftarrow$ `read`
    at interface A of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$

  // Valid epoch?
  Let e_{max} be max s.t. $\mathsf{Mem}^{sk_{(e_{max},i)},\mathtt{B}} \neq \bot$
  Let e_{min} be min s.t. $\mathsf{Mem}^{sk_{(e_{min},i)},\mathtt{B}} \neq \bot$
  if $ep = \bot \vee \neg(e_{min} \leq ep \leq e_{max})$ **then**
    $valid \leftarrow$ `false`

  // Decrypt
  call $sk \leftarrow$ `read`
    at interface B of $\mathsf{Mem}^{sk_{ep,i},\mathtt{B}}$
  if $sk \neq \bot \wedge valid$ **then**
    $m \leftarrow \text{HIBE.Dec}(sk, c)$

  // Update the keys
  if $valid \wedge m \neq \bot$ **then**
    $h \leftarrow \text{hash}(c, ep, ad)$
    for $k = ep, \ldots, e_{max}$ **do**
      call $sk \leftarrow$ `read`
        at int. B of $\mathsf{Mem}^{sk_{(k,i)},\mathtt{B}}$
      $sk' \leftarrow \text{HIBE.Kgen}(sk, h)$
      call $(\mathtt{write}, sk')$
        at int. B of $\mathsf{Mem}^{sk_{(k,i+1)},\mathtt{B}}$
  for $k = e_{min}, \ldots, e_{max}$ **do**
    call `erase`
      at interface B of $\mathsf{Mem}^{sk_{(k,i)},\mathtt{B}}$

  if $valid \wedge m \neq \bot$ **then return** (m, ad)
  else return $\bot$

Input: `isAvailable`
  call $succ \leftarrow$ `isAvailable`
    at interface B of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  return `true` iff $succ$ and there exists j s.t. $\mathsf{Mem}^{sk_{(j,i)},\mathtt{B}} \neq \bot$

Emulating Int. B **of** $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}, j \in [n]$

Input: $(\mathtt{send}, m, ad) \in \mathcal{M} \times \mathcal{AD}$
  require only called once & isAvailable

  // Number of received messages
  Let i be max s.t. $\mathsf{Mem}^{sk_{(j-1,i+1)},\mathtt{B}} \neq \bot$ if $j > 1$ and 0 otherwise.

  // Generate the (mpk,msk) pair
  call $r \leftarrow$ `sample` at int. B of $\mathsf{Rand}^{kg,\mathtt{B}}$
  $(mpk, msk) \leftarrow \text{HIBE.Setup}(r)$
  call $(\mathtt{write}, m, (mpk, i, ad))$
    at interface A of $\mathsf{Ch}^{k,\mathtt{B}\to\mathtt{A}}$
  call $(\mathtt{write}, msk)$
    at interface B of $\mathsf{Mem}^{sk_{(j,i+1)},\mathtt{B}}$
  return `ok`

Input: `isAvailable`
  call $succ \leftarrow$ `isAvailable`
    at interface B of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$
  return `true` iff $succ$ and $(j-1)$-st sent

Figure C.6: A formal description of the converter hibe-dec for the receiver.

C.2.2 Proof of Theorem 6.3.3

We require the HIBE scheme to be IND-CCA secure with the following (non-standard) additional properties: first, we require that decrypting a honestly generated ciphertext with an unauthorized decryption key (further down or in a different path in the hierarchy) not only decrypts to something unrelated, but fails to decrypts except with negligible probability.[1] Second, we require encryption to be truly randomized, that is, even given the secret key, no adversary can output a message-ciphertext pair such that a fresh encryption of the message results in that ciphertext, except with negligible probability.[2]

Proof. A formal description of the simulator $\boldsymbol{\sigma}_{\mathsf{hibe}}$ is presented in Figure C.7. The simulator initially samples all randomness, for the key generation and the encryption, and generates all the master key pairs. Using this, it is easy to simulate the randomness resources $\mathsf{Rand}^{kg_j,\mathtt{B}}$, $\mathsf{Rand}^{enc_i,\mathtt{A}}$, as well as the memories storing the master secret keys. Furthermore, simulating the channels for transporting the master public keys, and the memory $\mathsf{Mem}^{pk,\mathtt{A}}$ storing them, is trivial as well. The other memories $\mathsf{Mem}^{sk_{(j,i)},\mathtt{B}}$ and $\mathsf{Mem}^{tr_i,\mathtt{A}}$ storing the derived secret keys and the ciphertext hashes, respectively, can be easily simulated once we can consistently simulate the ciphertexts.

Observe that the simulator needs to generate ciphertexts in the following situations:

- The distinguisher explicitly queries the i-th ciphertext, or the hash thereof.
- The distinguisher asks for the i-th secret key of the j-th epoch: In this situation the simulator needs to generate all ciphertexts, up to the first injection, with which the receiver updated the j-th master secret key. After altering the keys for the j-th epoch with an injection, we know from our additional assumption that forwarding the correct ciphertext will cause a decryption error, so Eve needs to provide her own, which are then used to compute the secret keys.

[1] This can be achieved by including sufficient redundancy.

[2] For instance, IND-CCA allows the public-key to encode a message for which encryption is deterministic, as long as one cannot devise that message from the public key.

- The distinguisher injects the first message in an epoch. At this point, the secret key is still in sync and the simulator needs to decrypt under that key to determine the message.

To simulate the ciphertexts we follow the usual strategy: if the message is known to the simulator at the time of simulating the ciphertext, then he simply encrypts it. Otherwise, if only the length is known, he encrypts the string of zeros of the same length instead. By the CCA-security of the HIBE scheme, this is indistinguishable unless after encrypting the fake message the simulator has to reveal an earlier secret-key for that epoch (up to an injection) that allows for trivial decryption. Note that keys after an injection do not reveal anything about the encryptions, since the identities do not match.

Our interval-wise relaxation, however, means that we don't have to care about fake ciphertexts being exposed after each of the situation in which the simulator had to produce one. In addition, note that whenever the encryption randomness leaked, the simulator gets the real message immediately, due to our additional assumption in the real world, the simulator is allowed to fetch the message, without triggering an event, as soon as the channel becomes insecure. Hence, he can simply encrypt the correct message and avoid the commitment issue.

Finally, consider how delivering on the channels is handled by the simulator. First, observe that the simulator can process the channels in order, i.e, delay handling them until all previous ones have been handled, since Bob will only fetch them in order. To actually handle the i-th delivery request, the simulator proceeds as follows: firstly, he determines whether the ciphertext, epoch number, and associated data that Eve wants to deliver are the same as Alice sent. For the ciphertext we can use the following strategy:

- If Alice did not send her message yet, then she also did not sample the corresponding randomness. Hence, by our additional assumption we know that it won't be the same ciphertext.
- If Alice sent her message and Eve requests to forward it, it is trivially the same.
- If Alice sent her message, and either we already simulated the ciphertext—or the message is not secret and we can simply create the real one now—then we can simply compare.

- If Alice sent her message but it is still secret (in particular, the randomness did not leak yet) and Eve did not see the simulated ciphertext yet, then it is a different one, since again the randomness is unknown to Eve.

If the ciphertext matches, then we can also simply check the epoch number and associated data.

Once the simulator knows whether Eve's message matches Alice's, it knows whether Bob will update his secret keys with the correct identity. Hence, we especially known after processing each delivery request whether at the end Bob will still have the correct decryption key (without having to generate them) or whether the parties are now out of sync. Based on this information, the simulator proceeds as follows:

- If it is the same ciphertext, and Bob will use the correct decryption key (i.e., correct epoch and key of this epoch has been updated in sync), issue a forward command for the message. By correctness of the scheme, this simulates decryption correctly.

- If it is the same ciphertext, but Bob uses the wrong secret key, then issue an decryption error. By our stronger assumption, this simulates the real-world behavior.

- If it is a different ciphertext, the simulator generates the corresponding decryption key, performs the decryption, and either injects the message or triggers the decryption error, depending on the outcome.

It remains to argue that the simulator also works correctly if Eve tampered the transmission of the master public key, or the associated value indicating how many messages the receiver already obtained when creating that key. In this situation, all messages sent by Alice are treated as insecure and the simulator, thus, can simulate the actual ciphertexts. For delivery request, not that modifying either value will lead to Alice encrypting for the wrong key: either the master public key or the identity does not match. We can consider two situations: if the request happens after it is clear that Alice uses a wrong public key or identity, then the parties are treated as out of sync. If Eve forwards a ciphertext, then by our assumption, Bob's decryption will fail, which is what the simulator replicates. For newly injected ciphertexts, the simulator anyway always

replicates the correct decryption. If Eve request a delivery at which point it is unclear yet whether Alice uses the correct public key, then the ciphertext will be treated as an injection, and the simulator just replicates Bob's behavior of the real world. □

Simulator σ_{hibe}

Let HibeEncLen(ℓ, d) denote the function determining the ciphertext length based on the message length ℓ and the hierarchy level d.

Initialization

$C, C', AD', TR, MPK, MPK', MSK,$
$SK, E', R', InSync \leftarrow$ array init. to $\perp$
$(r^1_{kg}, \ldots, r^n_{kg}) \twoheadleftarrow \mathcal{R}^n$
$(r^1_{enc}, \ldots, r^n_{enc}) \twoheadleftarrow \mathcal{R}^n$
for $j \in [n]$ **do**
  $(MSK[j], MPK[j])$
    $\leftarrow$ HIBE.Kgen(r^j_{kg})

Emulating Int. E **of** $\mathsf{IMem}^{pk,\mathtt{A}}$

Input: read
  Let j max s.t. $\mathcal{E}^{\mathsf{received}}_{\mathsf{Ch}(j,\mathtt{B}\to\mathtt{A})}$ and
  $\neg\mathcal{E}^{\mathsf{received}}_{\mathsf{Ch}(j+1,\mathtt{B}\to\mathtt{A})}$
  if no such j exists **then**
    return $\perp$
  else
    return $(j, MPK'[j], R'[j])$

Emulating Int. E **of** $\mathsf{Mem}^{tr_i,\mathtt{A}}, i \in [n]$

Input: read
  if $\neg\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \vee \neg\mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Mem}(tr_i,\mathtt{A})}(\tilde{\mathcal{E}})$ **then**
    return $\perp$
  if $TR[i] = \perp$ **then**
    CREATETRANSCRIPT(i)
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(tr_i,\mathtt{A})}$
  return $TR[i]$

Emulating Int. E **of** $\mathsf{Mem}^{sk_{(j,i)},\mathtt{B}}$, $j \in [n], i \in [n+1]$

Input: read
  if $\mathcal{E}^{\mathsf{received}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \vee \mathcal{E}^{\mathsf{error}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})}$
    $\vee\neg\mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Mem}(sk_{(j,i)},\mathtt{B})}(\tilde{\mathcal{E}})$ **then**
    return $\perp$
  CREATESK(j, i)
  if $SK[j,i] \neq \perp$ **then**
    $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(sk_i,\mathtt{B})}$
  return $SK[j,i]$

Emulating Int E **of** $\mathsf{Rand}^{kg_j,\mathtt{B}}, j \in [n]$

Input: triggerLeaking
  if $\neg\mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Rand}(kg_j,\mathtt{B})}(\tilde{\mathcal{E}})$ **then return** $\perp$
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(kg_j,\mathtt{B})}$
  return ok

Input: getLeakage
  if $\neg\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(j,\mathtt{B}\to\mathtt{A})} \vee \neg\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(kg_j,\mathtt{B})}$ **then**
    return $\perp$
  return r^j_{kg}

Emulating Int. E of $\mathsf{Rand}^{enc_i,\mathtt{A}}$, $i \in [n]$

Input: `triggerLeaking`
  if $\neg \mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Rand}(enc_i,\mathtt{A})}(\bar{\mathcal{E}})$ **then return** $\bot$
  $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(enc_i,\mathtt{A})}$
  return `ok`

Input: `getLeakage`
  if $\neg\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \vee \neg\mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(enc_i,\mathtt{A})}$ **then**
    return $\bot$
  return r^i_{enc}

Emulating Int. E of $\mathsf{Ch}^{j,\mathtt{B}\to\mathtt{A}}$, $j \in [n]$

Input: `read`
  call $(m, ad) \leftarrow$ `read` at int. E of $\mathsf{Ch}^{k,\mathtt{B}\to\mathtt{A}}$
  if $(m, ad) = \bot$ **then return** $\bot$
  else
    $r \leftarrow$ Received(j)
    return $(m, (MPK[j], r, ad))$

Input: `readLength`
  call $(\ell, ad) \leftarrow$ `readLength` at interface E of $\mathsf{Ch}^{k,\mathtt{B}\to\mathtt{A}}$
  if $(\ell, ad) = \bot$ **then return** $\bot$
  else
    $r \leftarrow$ Received(j)
    return $(\ell, (MPK[j], r, ad))$

Input: $(\mathtt{error}, err, Overw, m, (mpk', r', ad), same)$
  $r \leftarrow$ Received(j)
  if $same = \mathtt{check} \wedge (mpk', r') \neq (MPK[j], r)$ **then**
    $same \leftarrow \mathtt{false}$
    $InSync[j, r] \leftarrow \mathtt{false}$
  call $(\mathtt{error}, err, Overw, m, ad, same)$ at interface E of $\mathsf{Ch}^{k,\mathtt{B}\to\mathtt{A}}$
  return `ok`

Input: $(\mathtt{deliver}, m, (mpk', r', ad), same)$
  $MPK'[j] \leftarrow mpk'$
  $R'[j] \leftarrow r'$
  $r \leftarrow$ Received(j)
  if $(mpk', r') = (MPK[j], r)$ **then**
    $InSync[j, r] \leftarrow \mathtt{true}$
  else
    $same \leftarrow \mathtt{false}$
    $InSync[j, r] \leftarrow \mathtt{false}$
  call $(\mathtt{deliver}, m, ad, same)$ at interface E of $\mathsf{Ch}^{k,\mathtt{B}\to\mathtt{A}}$
  return `ok`

Emulating Int. E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$, $i \in [n]$

Input: `read`
  if $\mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Chan}(i,\mathtt{A}\to\mathtt{B})} = \mathtt{false}$ **then**
    return $\bot$
  else if $\mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Chan}(i,\mathtt{A}\to\mathtt{B})} = \mathtt{true}$ **then**
    $\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Chan}(i,\mathtt{A}\to\mathtt{B})}$
  call $(\ell, ad) \leftarrow$ `readLength` at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  $e \leftarrow$ Epoch(i)
  if $C[i] = \bot$ **then**
    CreateCiphertext(i)
  return $(C[i], (e, ad))$

Input: `readLength`
  call $(\ell, ad) \leftarrow$ `readLength` at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  if $(\ell, ad) = \bot$ **then return** $\bot$
  $e \leftarrow$ Epoch(i)
  return $(\text{HibeEncLen}(\ell, i-1-R'[e]), (e, ad))$

Input: $(\mathtt{deliver}, c', (e', ad'), same)$
  if $c' = \mathtt{fwd} \wedge \neg\mathcal{E}^{\mathsf{sent}}_{\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}}$ **then**
    return $\bot$

  At this point, if delivery of $(i-1)$-st channel not called, then output `ok` immediately but delay processing until then.

  $(E'[i], AD'[i], C'[i]) \leftarrow (e', ad', c')$
  if $\neg$ValidEpoch(e', i) **then**
    call $(\mathtt{error}, \mathsf{dec\text{-}err}, \emptyset, same)$ at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
    return `ok`

  // Same ciphertext?
  call $(\ell, (e, ad)) \leftarrow$ `readLength` at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  if $(\ell, (e, ad)) = \bot$ **then**
    $sameC \leftarrow \mathtt{false}$
  else if $c' = \mathtt{fwd}$ **then**
    $sameC \leftarrow \mathtt{true}$
  else if $C[i] \neq \bot \vee \mathfrak{L}^{\mathsf{S}^{\mathsf{hibe}}}_{\mathsf{Chan}(i,\mathtt{A}\to\mathtt{B})}(\mathcal{E}) = \mathsf{silent}$ **then**
    if $C[i] = \bot$ **then**
      CreateCiphertext(i)
    $sameC \leftarrow (C[i] = c')$
  else
    $sameC \leftarrow \mathtt{false}$

(continued on next page)

Int $\mathtt{E}$ of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$ cont.

Input: `deliver` (continued.)

// Overall: injection?
$same' \leftarrow sameC \wedge (e', ad') = (e, ad)$
$InSync[e', i] \leftarrow InSync[e', i-1] \wedge same'$
if $\neg(same')$ **then** $same \leftarrow$ `false`

// Handle request
if $sameC$ **then**
  if $e = e' \wedge InSync[e', i-1]$ **then**
    call (`deliver`, `fwd`, ad', $same$) at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  else
    call (`error`, dec-err, $\emptyset$, $same$) at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
else
  if $SK[e', i] = \bot$ **then**
    CREATESK(e', i)
  $m' \leftarrow$ HIBE.Dec($SK[e', i], c'$)
  if $m' \neq \bot$ **then**
    call (`deliver`, m', ad', $same$) at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  else
    call (`error`, dec-err, $\emptyset$, $same$) at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
return `ok`

Input: (`error`, err, $Overw$, c', (e', ad'), $same$)
if $same =$ `check` **then**
  determine $same'$ as in for handling the `deliver` command above.
  $same \leftarrow same'$ // `true` or `false`
$Overw \leftarrow Overw \cup \{$dec-err$\}$
$m \twoheadleftarrow \mathcal{M}$ // ignored
call (`error`, err, $Overw$, m, ad', $same$) at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
return `ok`

Function Epoch(i)

$e = \max\left\{ j \;\middle|\; \mathcal{E}^{\text{received}}_{\mathsf{Ch}(j,\mathtt{B}\to\mathtt{A})} \prec \mathcal{E}^{\text{sent}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \right\}$
return e

Proc. ValidEpoch(e, i), $e, i \in [n]$

$rcv \leftarrow$ RECEIVED(e)
return $\mathcal{E}^{\text{received}}_{\mathsf{Ch}(i-1,\mathtt{A}\to\mathtt{B})} \wedge \mathcal{E}^{\text{sent}}_{\mathsf{Ch}(e,\mathtt{B}\to\mathtt{A})} \wedge rcv < i \wedge E'[i-1] \leq e$

Function Received(j)

// Number of msg. Bob received when sending the j-th msg.
$r = \max\left(\{i \in [n] \mid \mathcal{E}^{\text{received}}_{\mathsf{Ch}(i,\mathtt{A}\to\mathtt{B})} \prec \mathcal{E}^{\text{sent}}_{\mathsf{Ch}(j,\mathtt{B}\to\mathtt{A})}\} \cup \{0\}\right)$
return r

Proc. CreateCiphertext(i), $i \in [n]$

$e \leftarrow$ EPOCH(i)
$id \leftarrow ()$
for $k = (R'[e]+1), \ldots, (i-1)$ **do**
  if $TR[k] = \bot$ **then**
    CREATETRANSCRIPT(k)
  $id \leftarrow id \parallel TR[k]$
call $(m, ad) \leftarrow$ `read` at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
if $(m, ad) = \bot$ **then**
  call $(\ell, ad) \leftarrow$ `readLength` at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
  $m \leftarrow 0^{\ell}$
$C[i] \leftarrow$ HIBE.Enc($MPK'[e], id, m; r^{i}_{enc}$)

Proc. CreateTranscript(i), $i \in [n]$

if $C[i] = \bot$ **then**
  CREATECIPHERTEXT(i)
call $(\ell, ad) \leftarrow$ `readLenght` at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
$TR[i] \leftarrow$ hash($C[i]$, EPOCH(i), ad)

Procedure CreateSK(j, i)

$rcv \leftarrow$ RECEIVED(j)
if $SK[j, i] \neq \bot \vee \neg$VALIDEPOCH($j, i$) **then**
  return
else if $i = rcv + 1$ **then**
  $SK[j, i] \leftarrow MSK[j]$
else
  CREATESK($j, i-1$)
  if $C'[i-1] =$ `fwd` **then**
    if $C[i-1] = \bot$ **then**
      CREATECIPHERTEXT($i-1$)
    $C'[i-1] \leftarrow C[i-1]$
  $h \leftarrow$ hash(C'[i-1], E'[i-1], AD'[i-1])
  $SK[j, i] \leftarrow$ HIBE.Kgen($SK[j, i\text{-}1], h$)

Figure C.7: A formal description of the simulator used for Theorem 6.3.3.

C.3 Details of Section 6.5

C.3.1 Simulator from Theorem 6.5.1

In this section, we provide a description of the simulator used in the proof for Theorem 6.5.1. The simulator $\boldsymbol{\sigma}_{\mathsf{CKA}}$, as depicted in Figure C.8, has to emulate the state memories and randomness resources. Moreover, it has to adjust the channel interfaces (where the CKA-values (e, T) are not present in the ideal world) and map the key resources to the corresponding memory resources in the real world.

Simulator σ_{CKA}

Initialization

Sample random values $(r_1, \ldots, r_{2n})$
Execute the CKA scheme for $2n$ epochs (with the above randomness), and for each epoch e store the resulting values in $\gamma^{\mathtt{A}}[e], \gamma^{\mathtt{B}}[e], I[e]$ and $T[e]$. Initialize the empty dictionary Tr, storing all values e and T delivered in authenticated data.

Emulating Interface $\mathtt{E}$ of $\mathsf{Rand}^{e,\mathtt{A}}$, odd $e \in [2n]$

Input: `triggerLeaking`
if $\neg \mathfrak{L}^{\mathsf{R}^{\mathsf{hibe}}}_{\mathsf{Rand}(kg_j,\mathtt{B})}(\tilde{\mathcal{E}})$ **then return** $\bot$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(e,\mathtt{A})}$
return `ok`

Input: `getLeakage`
Using the event history, determine the epoch $e_{\mathtt{A}}$, such that Alice is currently in $e_{\mathtt{A}}$.
if $e_{\mathtt{A}} < e \vee \neg \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Rand}(e,\mathtt{A})}$ **then**
  return $\bot$
return r_e

Emulating Interface $\mathtt{E}$ of $\mathsf{IMem}^{ep,\mathtt{A}}$

Input: `read`
Using the event history, determine the epoch $e_{\mathtt{A}}$, such that Alice is currently in $e_{\mathtt{A}}$.
return $e_{\mathtt{A}}$

Emulating Interface $\mathtt{E}$ of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$, $i \in [n]$

Input: `read`
If $\mathfrak{L}^{\mathsf{R}_{\mathsf{CKA}}}_{\mathsf{Chan}(i,\mathtt{A}\to\mathtt{B})} = \mathtt{false}$, **return** $\bot$.
call $(m, ad) \leftarrow$ `read`
  at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
Determine the i-th message's epoch e from the event history.
return $(m, (ad, e, T[e]))$

Input: `readLength`
call $(\ell, ad) \leftarrow$ `read`
  at interface E of $\mathsf{Ch}^{i,\mathtt{A}\to\mathtt{B}}$
Determine the i-th message's epoch e from the event history.
return $(\ell, (ad, e, T[e]))$

Input: $(\mathtt{deliver}, m, (ad', e', T'), same')$
require only called once
$\mathsf{Tr}[i, \mathtt{B}] \leftarrow (e', T')$
if $T[e'] \neq T'$ **then**
  Rerun CKA scheme from e' on (with original randomness) and overwrite the respective values in $\gamma^{\mathtt{A}}[e], \gamma^{\mathtt{B}}[e], I[e]$ and $T[e]$.
call $(\mathtt{inject}, inj(\mathsf{Tr}, e, \mathtt{B}, \cdot), \mathtt{B})$
  at interface E of $\mathsf{Key}^{e,(\mathtt{A},\mathtt{B})}$
Forward the command (without e' and T') to the ideal channel
return `ok`

Input: $(\mathtt{error}, err, O, m, (ad', e', T'), same')$
Forward the input (without e' and T') to the ideal channel
return `ok`.

Emulating Interface $\mathtt{E}$ **of** $\mathsf{Mem}^{st_{e,\mathtt{A}}}$**,** $e \in \{0, \ldots, 2n\}$ **(**including CKA-Setup**)**

Input: `read`

Using the event history, determine the epoch $e_{\mathtt{A}}$ Alice is currently in.
if $e \neq e_{\mathtt{A}}$ **then return** $\perp$
$\mathcal{E} \xleftarrow{+} \mathcal{E}^{\mathsf{leaked}}_{\mathsf{Mem}(st_{e,\mathtt{A}})}$
if e is even **then**
  return $\gamma^{\mathtt{A}}[e]$
else
  return $(\gamma^{\mathtt{A}}[e], T[e])$

Function $inj(\mathsf{Tr}, e', \gamma^{\mathtt{A}}, \gamma^{\mathtt{B}}, I, \mathtt{B}, \mathcal{E})$

With the help the event history $\mathcal{E}$, the transcript Tr and the values $\gamma^{\mathtt{A}}, \gamma^{\mathtt{B}}, I$, run Bob's protocol and determine the key k he outputs in epoch e' (set $k = \perp$ if this key has not been produced yet). Output k.

Emulating Interface $\mathtt{E}$ **of** $\mathsf{OMem}^{key_{e,\mathtt{A}}}$**,** $e \in [2n]$

Input: `read`

Using the event history, determine the epoch $e_{\mathtt{A}}$, such that Alice is currently in $e_{\mathtt{A}}$.
if $e > e_{\mathtt{A}}$ **then return** $\perp$
call $k \leftarrow (\mathtt{read}, \mathtt{A})$
  at interface E of $\mathsf{Key}^{e,(\mathtt{A},\mathtt{B})}$
return k

Figure C.8: The simulator for the CKA protocol, depicting the interfaces corresponding to Alice's resources. The interface for Bob's resources are analogous.

Bibliography

[ACD19] J. Alwen, S. Coretti, and Y. Dodis, "The double ratchet: Security notions, proofs, and modularization for the signal protocol", in *Advances in Cryptology — EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds., Berlin, Heidelberg: Springer International Publishing, 2019.

[BDHK06] M. Backes, M. Dürmuth, D. Hofheinz, and R. Küsters, "Conditional reactive simulatability", in *Computer Security — ESORICS 2006*, D. Gollmann, J. Meier, and A. Sabelfeld, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 424–443.

[BPW07] M. Backes, B. Pfitzmann, and M. Waidner, "The reactive simulatability (rsim) framework for asynchronous systems", *Information and Computation*, vol. 205, no. 12, pp. 1685–1720, 2007. DOI: https://doi.org/10.1016/j.ic.2007.05.002.

[BHK13a] M. Bellare, V. T. Hoang, and S. Keelveedhi, *Instantiating random oracles via uces*, Cryptology ePrint Archive, Report 2013/424 (preliminary version June 2013), https://eprint.iacr.org/2013/424, Jun. 2013.

[BHK13b] ——, "Instantiating Random Oracles via UCEs", in *Advances in Cryptology — CRYPTO 2013*, Springer Berlin Heidelberg, Aug. 2013, pp. 398–415.

[BHK14] ——, "Cryptography from compression functions: The UCE bridge to the ROM", in *Advances in Cryptology —*

CRYPTO 2014. Springer Berlin Heidelberg, 2014, pp. 169–187.

[BR93] M. Bellare and P. Rogaway, "Random oracles are practical: A Paradigm for Designing Efficient Protocols", in *1st ACM Conference on Computer and Communications Security — CCS 93*, ACM Press, 1993, pp. 62–73.

[BSJ+17] M. Bellare, A. C. Singh, J. Jaeger, M. Nyayapati, and I. Stepanovs, "Ratcheted encryption and key exchange: The security of messaging", in *Advances in Cryptology — CRYPTO 2017*, 2017, pp. 619–650.

[Blu83] M. Blum, "Coin flipping by telephone a protocol for solving impossible problems", *SIGACT News*, vol. 15, no. 1, pp. 23–27, Jan. 1983. DOI: 10.1145/1008908.1008911.

[BF01] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing", in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 213–229.

[BK12] Z. Brakerski and Y. Kalai, "A parallel repetition theorem for leakage resilience", in *Theory of Cryptography — TCC 2012*, Springer Berlin Heidelberg, 2012, pp. 248–265.

[BDH+17] B. Broadnax, N. Döttling, G. Hartung, J. Müller-Quade, and M. Nagel, "Concurrently composable security with shielded super-polynomial simulators", in *Advances in Cryptology — EUROCRYPT 2017*, J.-S. Coron and J. B. Nielsen, Eds., Cham: Springer International Publishing, 2017, pp. 351–381.

[BFM14] C. Brzuska, P. Farshim, and A. Mittelbach, "Indistinguishability Obfuscation and UCEs: The Case of Computationally Unpredictable Sources", in *Advances in Cryptology — CRYPTO 2014*, Springer Berlin Heidelberg, 2014, pp. 188–205.

[BM14] C. Brzuska and A. Mittelbach, "Using Indistinguishability Obfuscation via UCEs", in *Advances in Cryptology — ASIACRYPT 2014*, Springer Berlin Heidelberg, 2014, pp. 122–141.

[CEK+16] J. Camenisch, R. R. Enderlein, S. Krenn, R. Küsters, and D. Rausch, "Universal composition with responsive environments", in *Advances in Cryptology — ASIACRYPT 2016*, J. H. Cheon and T. Takagi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 807–840.

[CKKR19] J. Camenisch, S. Krenn, R. Küsters, and D. Rausch, "Iuc: Flexible universal composability made simple", in *Advances in Cryptology — ASIACRYPT 2019*, S. D. Galbraith and S. Moriai, Eds., Cham: Springer International Publishing, 2019, pp. 191–221.

[Can01] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols", in *42nd IEEE Symposium on Foundations of Computer Science — FOCS 2001*, IEEE Computer Society, 2001, pp. 136–145.

[CDPW07] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup", in *Theory of Cryptography*, S. P. Vadhan, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 61–85.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation", in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96, Philadelphia, Pennsylvania, USA: ACM, 1996, pp. 639–648. DOI: 10.1145/237814.238015.

[CF01] R. Canetti and M. Fischlin, "Universally composable commitments", in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 19–40.

[CGH04] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited", *Journal of the ACM*, vol. 51, no. 4, pp. 557–594, Jul. 2004.

[CHK03] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme", in *Advances in Cryptology — EUROCRYPT 2003*, E. Biham, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 255–271.

[CHK05] ——, "Adaptively-secure, non-interactive public-key encryption", in *TCC 2005: 2nd Theory of Cryptography Conference*, J. Kilian, Ed., ser. Springer, Heidelberg, vol. 3378, Springer, Heidelberg, 2005, pp. 150–168.

[CK02] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels", in *Advances in Cryptology — EUROCRYPT 2002*, L. R. Knudsen, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 337–351.

[CR03] R. Canetti and T. Rabin, "Universal composition with joint state", in *Advances in Cryptology — CRYPTO 2003*, D. Boneh, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 265–281.

[CCD+17] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A Formal Security Analysis of the Signal Messaging Protocol", *2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017*, pp. 451–466, 2017.

[DFR+07] I. B. Damgård, S. Fehr, R. Renner, L. Salvail, and C. Schaffner, "A Tight High-Order Entropic Quantum Uncertainty Relation with Applications", in *Advances in Cryptology — CRYPTO 2007*, Springer Berlin Heidelberg, 2007, pp. 360–378.

[DGHM13] G. Demay, P. Gaži, M. Hirt, and U. Maurer, "Resource-Restricted Indifferentiability", in *Advances in Cryptology — EUROCRYPT 2013*, Springer Berlin Heidelberg, 2013, pp. 664–683.

[DGMT17] G. Demay, P. Gaži, U. Maurer, and B. Tackmann, "Per-session security: Password-based cryptography revisited", in *Computer Security — ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds., Cham: Springer International Publishing, 2017, pp. 408–426.

[DV18] F. B. Durak and S. Vaudenay, *Bidirectional asynchronous ratcheted key agreement with linear complexity*, Cryptology ePrint Archive, Report 2018/889, `https://eprint.iacr.org/2018/889`, 2018.

[FM16] P. Farshim and A. Mittelbach, "Modeling Random Oracles Under Unpredictable Queries", in *Fast Software Encryption — FSE 2016*, Springer Berlin Heidelberg, 2016, pp. 453–473.

[HMM15] D. Hofheinz, C. Matt, and U. Maurer, "Idealizing identity-based encryption", in *Advances in Cryptology – ASIACRYPT 2015*, T. Iwata and J. H. Cheon, Eds., ser. Lecture Notes in Computer Science, vol. 9452, Springer Berlin Heidelberg, 2015, pp. 495–520.

[HS15] D. Hofheinz and V. Shoup, "Gnuc: A new universal composability framework", *Journal of Cryptology*, vol. 28, no. 3, pp. 423–508, Jul. 2015. DOI: 10.1007/s00145-013-9160-y.

[JS18] J. Jaeger and I. Stepanovs, "Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging", in *Advances in Cryptology — CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds., Springer, 2018, pp. 33–62.

[JM18] D. Jost and U. Maurer, "Security definitions for hash functions: Combining uce and indifferentiability", in *Security and Cryptography for Networks*, D. Catalano and R. De Prisco, Eds., Springer International Publishing, 2018, pp. 83–101.

[JM20] ——, "Overcoming impossibility results in composable security using interval-wise guarantees", in *Advances in Cryptology — CRYPTO 2020*, (to appear), Cham: Springer International Publishing, 2020.

[JMM19a] D. Jost, U. Maurer, and M. Marta, "Efficient ratcheting: Almost-optimal guarantees for secure messaging", in *Advances in Cryptology — EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds., Berlin, Heidelberg: Springer International Publishing, 2019.

[JMM19b] D. Jost, U. Maurer, and M. Mularczyk, "A unified and composable take on ratcheting", in *Theory of Cryptography — TCC 2019*, D. Hofheinz and A. Rosen, Eds., Cham: Springer International Publishing, Dec. 2019, pp. 180–210.

[Kra01] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure is ssl?)", in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 310–331.

[KTR13] R. Kuesters, M. Tuengerthal, and D. Rausch, *The iitm model: A simple and expressive model for universal composability*, Cryptology ePrint Archive, Report 2013/025, `https://eprint.iacr.org/2013/025`, 2013.

[Mau11] U. Maurer, "Constructive Cryptography–A New Paradigm for Security Definitions and Proofs", in *Theory of Security and Applications — TOSCA 2011*, Springer Berlin Heidelberg, 2011, pp. 33–56.

[MRH04] U. Maurer, R. Renner, and C. Holenstein, "Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology", in *Theory of Cryptography — TCC 2004*, Springer Berlin Heidelberg, 2004, pp. 21–39.

[Mau02] U. Maurer, "Indistinguishability of random systems", in *Advances in Cryptology — EUROCRYPT 2002*, L. R. Knudsen, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 110–132.

[Mau13] ——, "Conditional equivalence of random systems and indistinguishability proofs", in *2013 IEEE International Symposium on Information Theory*, IEEE, 2013, pp. 3150–3154.

[MPR07] U. Maurer, K. Pietrzak, and R. Renner, "Indistinguishability amplification", in *Advances in Cryptology — CRYPTO 2007*, A. Menezes, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 130–149.

[MR11] U. Maurer and R. Renner, "Abstract cryptography", in *In Innovations in Computer Science — ICS 2011*, Tsinghua University, 2011, pp. 1–21.

[MR16] ——, "From Indifferentiability to Constructive Cryptography (and Back)", in *Theory of Cryptography — TCC 2016*, Springer Berlin Heidelberg, 2016, pp. 3–24.

[MT10] U. Maurer and B. Tackmann, "On the soundness of authenticate-then-encrypt: Formalizing the malleability of symmetric encryption", in *Proceedings of the 17th ACM Conference on Computer and Communication Security*, A. D. Keromytis and V. Shmatikov, Eds., ACM, ACM, Oct. 2010, pp. 505–515.

[Mit14] A. Mittelbach, "Salvaging Indifferentiability in a Multi-stage Setting", in *Advances in Cryptology — EUROCRYPT 2014*, Springer Berlin Heidelberg, 2014, pp. 603–621.

[Nie02a] J. B. Nielsen, "Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case", in *Advances in Cryptology — CRYPTO 2002*, M. Yung, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 111–126.

[Nie02b] ——, "Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case", in *Advances in Cryptology — CRYPTO*, M. Yung, Ed., ser. Lecture Notes in Computer Science, vol. 2442, Springer, Heidelberg, 2002, pp. 111–126. DOI: 10.1007/3-540-45708-9_8.

[OWS17] *Open Whisper Systems. Signal protocol library for java/android.* GitHub repository, Accessed: 2018-10-01, 2017. [Online]. Available: https://github.com/WhisperSystems/libsignal-protocol-java.

[Pas03] R. Pass, "Simulation in quasi-polynomial time, and its application to protocol composition", in *Advances in Cryptology — EUROCRYPT 2003*, E. Biham, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 160–176.

[PW01] B. Pfitzmann and M. Waidner, "A model for asynchronous reactive systems and its application to secure message transmission", in *Proceedings 2001 IEEE Symposium on Security and Privacy — S&P 2001*, May 2001, pp. 184–200. DOI: 10.1109/SECPRI.2001.924298.

[Poe18] Poettering, Bertram and Rösler, Paul, "Towards Bidirectional Ratcheted Key Exchange", in *Advances in Cryptology — CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds., Cham: Springer International Publishing, 2018, pp. 3–32.

[PS04] M. Prabhakaran and A. Sahai, "New notions of security: Achieving universal composability without trusted setup", in *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '04, Chicago, IL, USA: ACM, 2004, pp. 242–251. DOI: 10.1145/1007352.1007394.

[RSS11] T. Ristenpart, H. Shacham, and T. Shrimpton, "Careful with Composition: Limitations of Indifferentiability and Universal Composability", in *Advances in Cryptology — EUROCRYPT 2011*, Springer Berlin Heidelberg, 2011, pp. 487–506.

[ST17] P. Soni and S. Tessaro, "Public-Seed Pseudorandom Permutations", in *Advances in Cryptology — EUROCRYPT 2017*, Springer International Publishing, 2017, pp. 412–441.

[Wul07] J. Wullschleger, "Oblivious-Transfer Amplification", in *Advances in Cryptology — EUROCRYPT 2007*, Springer Berlin Heidelberg, 2007, pp. 555–572.