Doctoral Thesis

Cryptographic Constructions of Randomness Resources

Author(s): Demay, Grégory

Publication Date: 2016

Permanent Link: https://doi.org/10.3929/ethz-a-010692296

Rights / License: In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection. For more information please consult the Terms of use.
Cryptographic Constructions of Randomness Resources

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

Grégory Jean Dominique Demay
Ingénieur diplômé de Télécom Bretagne

born on January 15, 1987
citizen of the French Republic

accepted on the recommendation of

Prof. Dr. Ueli Maurer, examiner
Prof. Dr. David Basin, co-examiner
Prof. Dr. Dennis Hofheinz, co-examiner

2016
First of all, I would like to express my sincere gratitude to my supervisor Ueli Maurer. His dedication to research and his unique abstract view on the field of Cryptography has lead to many enjoyable and fruitful discussions.

I would also like to thank Dennis Hofheinz and David Basin for agreeing to serve as co-examiners on my committee and for their valuable comments.

My stay in the Information Security and Cryptography Research Group has been made particularly pleasant thanks to its former and current members: Divesh Aggarwal, Joël Alwen, Christian Badertscher, Kfir Barhum, Sandro Coretti, Maria Dubovitskaya, Robert Enderlein, Gian Pietro Farina, Peter Gaži, Martin Hirt, Daniel Jost, Simon Knellwolf, Chen-Da Liu Zhang, Julian Loss, Christoph Lucas, Christian Matt, Arpita Patra, Pavel Raykov, Gregor Seiler, Björn Tackmann, Daniel Tschudi and Vassilis Zikas. I am particularly indebted to Peter, Björn and Martin for collaboration that led to some of the results presented in this thesis. I would also like to thank Sandro, Kfir and Pavel for many unforgettable moments, as well as my other office mates Robert and Christian B. who managed to endure me. The latter had in addition the difficult task of correcting my plethora of German mistakes. My thank you also goes to our group and department secretaries Beate Bernhard, Claudia Günthart and Denise Spicher who ensured the smoothness of any administrative process.

Most of this would not have been possible without the love and support of my parents, Jean and Dominique, and of my dearest friends, Edouard P., François, Pierre, Nicolas, Maxime, Edouard J., Mélanie and Aude. Life in Zürich would have been a lot less entertaining without a Spanish touch, brought by my long term flatmate Adrián and ma douce colombe Paloma. Merci à tous!

Finally, I am grateful to the Zurich Information Security and Privacy Center (ZISC) for supporting this work.
Abstract

Randomness is one of the most fundamental resources in cryptography and is central to most cryptographic schemes, ranging from message encryption to multi-party computation. Following the paradigm of abstract and constructive cryptography introduced by Maurer and Renner (ICS 2011), we understand a cryptographic scheme (e.g., one-time pad) as a way to construct, in a well-defined sense, a resource that formalizes the desired security guarantees (e.g., a secure channel guarantees authenticity and confidentiality) from one or more resources that are assumed to be available (e.g., a shared secret key and an authenticated channel).

This thesis focuses on constructions involving different randomness resources, which can be broadly classified into two categories. The first category consists of showing whether some resources deemed of interest can (or cannot) be constructed from a particular randomness resource that is assumed to be available. The second category consists instead of targeting a particular desired randomness resource and then showing how (if at all) it can be constructed. The composition theorem of constructive cryptography ensures that all constructions shown in this thesis are composable and can therefore be directly reused as building blocks in other applications, an important property when dealing with ubiquitous low-level primitives such as randomness.

Password-based cryptography in the random oracle model. The first and main part of this thesis addresses constructions where a resource providing cryptographic keys derived from hashing (arbitrarily distributed) passwords is assumed to be available. Human-memorable passwords represent one of the most widely deployed security mechanisms in practice and it is therefore extremely important that we have a precise understanding of the security guarantees that passwords provide for multiple sessions (where one session is protected by one password).
The first challenge is to devise definitions, since cryptographic security is usually defined as some form of guarantee that holds except when a bad event with negligible probability occurs, and nothing is guaranteed in that case. However, in settings where sessions are protected by (possibly weak) passwords instead of unguessable cryptographic keys, one cannot resign on all security guarantees as soon as a single password is guessed. In this case, a *per-session* guarantee is desired: any session for which the password has not been guessed remains secure, independently of whether other sessions have been compromised. In particular, a user with a very strong password enjoys the full security guarantees of an analysis in which passwords are replaced by uniform cryptographic keys.

Our first contribution is therefore to provide a new, general technique for stating security guarantees that degrade gracefully and which could not be expressed with existing formalisms. Our method is simple and does not require new security definitions. Our second contribution is then to apply our approach to revisit the analysis of password-based message authentication and of password-based encryption, investigating whether they provide strong per-session guarantees. Our results also resolve an open problem stated by Bellare, Ristenpart and Tessaro (CRYPTO 2012).

Passwords are often poorly chosen, falling within a set small enough to be exhaustible. A useful technique to impede such brute-force attacks is to increase the computational complexity of evaluating a hash function, both for the honest users as well as for an adversary. A natural approach is to iterate a hash function $c$ times, for some parameter $c$, in the hope that any query to the scheme requires $c$ evaluations of the underlying hash function. However, results by Dodis et al. (CRYPTO 2012) imply that plain iteration falls short of achieving this goal, and designing schemes which provably have such a desirable property remained an open problem.

As our third contribution, we formalize explicitly what it means for a given scheme to amplify the query complexity of a hash function. In the random oracle model, the goal of a secure *query-complexity amplifier* scheme is captured as constructing, from a random oracle allowing $R$ queries (for the adversary), a random oracle provably allowing only $\mathcal{r} < R$ queries. Turned around, this means that making $\mathcal{r}$ queries to the scheme requires at least $R$ queries to the actual random oracle. Our fourth contribution is then to propose a new scheme, called collision-free iteration, that provably achieves $c$-fold query-complexity amplification for both the honest parties and the adversary, for any fixed parameter $c$. 
Common randomness between two distrustful parties. The second part of this thesis is devoted to constructions where the desired resource must provide some form of common randomness. Common randomness is prevalent in non-interactive zero knowledge proofs and is a necessary prerequisite in any composable protocol for multi-party computation.

Since common randomness is such a valuable resource, we introduce the task of common randomness amplification, where two distrustful parties are given some initial common randomness and by exchanging messages with one another wish to construct a new common randomness resource with higher Shannon entropy than what they initially shared. Our next contribution is then to show that common randomness amplification is information-theoretically impossible.

Finally, we consider common randomness in its simplest form: a coin tossing resource outputting the same random bit to all parties. Unfortunately, it is well-known that such a coin tossing resource cannot be constructed whenever a majority of players is dishonest. Our last contribution is therefore to introduce the notion of an unfair coin tossing resource in order to understand what coin tossing protocols really achieve in the setting of two distrustful parties.
Résumé

L’aléatoire est une ressource fondamentale en cryptographie et est centrale dans la plupart de ses applications, du chiffrement de messages aux calculs multipartites. En suivant le paradigme de cryptographie abstraite et constructive de Maurer et Renner (ICS 2011), un protocole cryptographique (e.g., le chiffrement de Vernam) est vu comme une façon de construire, dans un sens bien précis, une ressource qui formalise les garanties de sécurité désirées (e.g., un canal de communication sécurisé garantit authenticité et confidentialité) à partir d’une ou plusieurs ressources qui sont supposées être disponibles (e.g., une clé partagée secrète et un canal de communication authentifié).

Cette thèse se concentre sur des constructions impliquant différentes ressources d’aléatoire, qui peuvent être classifiées en deux grandes catégories. La première catégorie consiste à montrer si certaines ressources dignes d’intérêt peuvent (ou ne peuvent pas) être construites à partir d’une ressource d’aléatoire spécifique supposée disponible. La deuxième catégorie consiste au contraire à cibler une ressource d’aléatoire en particulier et à montrer comment elle peut être construite. Le théorème de composition de la cryptographie constructive garantit que toutes les constructions montrées dans cette thèse sont composables et peuvent donc être directement réutilisées comme des composants de base dans d’autres applications, un point crucial quand on traite de primitives cryptographiques qui sont à la fois omniprésentes et de bas niveau comme l’aléatoire.

Cryptographie basée sur des mots de passe dans le modèle de l’oracle aléatoire. La première et principale partie de cette thèse a pour objet des constructions où une ressource qui fournit des clés cryptographiques provenant du hachage de mots de passe (distribués de manière arbitraire) est supposée disponible. Let mots de passe mémorisables par des humains représentent un des mécanismes de sécurité le plus souvent
déployé en pratique et conséquemment il est extrêmement important de comprendre précisément les garanties de sécurité qu’offrent les mots de passe pour plusieurs sessions (où une session est protégée par un mot de passe).

Le premier défi est d’ordre définitionnel puisque la sécurité cryptographique est typiquement définie comme une forme de garantie qui a toujours lieu excepté dans le cas où un mauvais événement apparaît avec une probabilité négligeable, cas dans lequel aucune garantie n’ait donnée. Cependant, dans un scénario où des sessions sont protégées par des mots de passes (potentiellement faibles) en lieu de clé cryptographiques réellement aléatoires, abandonner toute notion de sécurité dès qu’un seul mot de passe est deviné serait inacceptable. Dans ce cas, une garantie par session est préférable, c’est-à-dire que n’importe quelle session dont le mot de passe n’a pas été deviné reste sécurisée, indépendamment du degré de compromission d’autres sessions. En particulier, un utilisateur avec un très fort mot de passe bénéficie des pleines garanties de sécurité que lui donnerait une analyse dans laquelle les mots de passe seraient remplacés par des clés cryptographiques totalement aléatoires.

Par conséquent, notre première contribution est de fournir une nouvelle et générale technique pour formuler des garanties de sécurité qui puissent être gracieusement dégradées et qui ne pouvaient pas être exprimées avec les formalismes existants. Notre méthode est simple et ne nécessite pas de nouvelles définitions de sécurité. Notre seconde contribution est ensuite d’appliquer notre procédé pour revisiter l’analyse de l’authentification de message basée sur des mots de passe et du chiffrement basé sur des mots de passe, afin de chercher à savoir s’ils fournissent de fortes garanties de sécurité par session. Nos résultats résolvent un problème ouvert par Bellare, Ristenpart et Tessaro (CRYPTO 2012).

Les mots de passe sont souvent mal choisis et appartiennent à un ensemble suffisamment petit pour pouvoir être complètement parcouru. Une technique utile pour entraver une telle attaque par force brute est d’augmenter la complexité informatique d’évaluation d’une fonction de hachage, pour les utilisateurs honnêtes comme pour un adversaire. Une première approche serait d’itérer une fonction de hachage $c$ fois, pour un paramètre $c$, dans l’espoir qu’une requête faite à ce système doive effectuer $c$ évaluations de la fonction de hachage sous-jacente. Malheureusement, les résultats de Dodis et al. (CRYPTO 2012) montrent que la simple itération ne peut pas réaliser ce but et concevoir un système de hachage qui a cette propriété attriante est resté un problème ouvert.
Notre troisième contribution est donc de formaliser de manière explicite ce que signifie pour un système de hachage d’augmenter la complexité d’évaluation d’une fonction de hachage. Dans le modèle de l’oracle aléatoire, le but d’un amplificateur de la complexité d’évaluation est de construire, à partir d’un oracle aléatoire permettant \( R \) requêtes (de l’adversaire), un oracle aléatoire permettant seulement \( n < R \) requêtes. Autrement dit, cela signifie que pour pouvoir faire \( n \) requêtes à l’amplificateur, \( R \) requêtes à l’oracle aléatoire sous-jacent sont nécessaires. Notre quatrième contribution est alors de proposer un nouveau système de hachage, appelé itération sans collision, qui réussit démonstrablement à amplifier par un facteur \( c \) la complexité d’évaluation pour les partis honnêtes et l’adversaire, pour tout paramètre fixe \( c \).

Aléatoire commun entre deux partis sans confiance mutuelle. La seconde partie de cette thèse est à propos de constructions où la ressource désirée doit fournir une sorte d’aléatoire commun. L’aléatoire commun est très répandu dans les preuves non interactives à divulgation nulle de connaissance et est nécessaire pour assurer la composition des protocoles de calculs multipartites.

Étant donné que l’aléatoire commun est une ressource de grande valeur, nous introduisons la notion d’amplification d’aléatoire commun, où deux partis qui ne se font pas confiance ont au début une certaine quantité d’aléatoire commun et en échangeant des messages entre l’un et l’autre souhaitent construire une nouvelle ressource d’aléatoire commun avec une entropie de Shannon plus grande que ce qu’ils avaient initialement. Notre contribution suivante est de montrer que l’amplification d’aléatoire commun est information-théoriquement impossible.

Finalement, nous considérons l’aléatoire commun dans sa forme la plus simple : une ressource de pile ou face qui donne aux deux partis le même bit d’aléatoire. Malheureusement, il est connu qu’une telle ressource de pile ou face ne peut pas être construite quand la majorité des partis est déshonnête. Notre dernière contribution est donc d’introduire la notion d’injuste ressource de pile ou face afin de comprendre ce que les protocoles de pile ou face accomplissent réellement quand deux partis ne se font pas mutuellement confiance.
### Contents

2.4.4 Multiple-Interface Random Systems .......................... 36  
2.5 Three Settings ...................................................... 39  
  2.5.1 The (Alice, Bob, Eve)-Setting ................................. 40  
  2.5.2 Indifferentiability .............................................. 41  
  2.5.3 Two Distrustful Parties ....................................... 43  
2.6 Symmetric Cryptographic Primitives .............................. 44  
  2.6.1 The Random Oracle Model ..................................... 44  
  2.6.2 MAC Schemes .................................................. 45  
  2.6.3 Encryption Schemes ............................................ 45

I Password-Based Cryptography in the Random Oracle Model 47

3 Per-Session Security 49  
  3.1 Transformable Systems ........................................... 49  
    3.1.1 Core Systems and Triggers .................................. 50  
    3.1.2 Downgradable Resources .................................... 51  
  3.2 Password-Based Key Derivation .................................. 54  
  3.3 Password-Based Message Authentication ........................ 55  
  3.4 Password-Based Encryption ...................................... 62  
    3.4.1 PBE for a Single Session .................................... 63  
    3.4.2 General Impossibility of PBE ............................... 65  
    3.4.3 PBE with Local Assumptions ................................ 70  
    3.4.4 Salting and Per-Session Security ......................... 77  
    3.4.5 On the Per-Session Security of PBE from PKCS #5 ...... 79  
  3.5 Conclusion of this Chapter ...................................... 84

4 Query-Restricted Systems 87  
  4.1 Parameterized Constructions ..................................... 87  
  4.2 CE Versus NA .................................................... 89  
    4.2.1 Distinguishing Restricted Systems .......................... 90  
    4.2.2 CE and Disjunction of Games ................................ 92  
    4.2.3 NA and Disjunction of Games ................................ 93  
    4.2.4 CE is Stronger ................................................ 95  
  4.3 Parameterized Constructions from CE ............................ 96  
    4.3.1 Single Conditional Equivalence Statement .................. 96  
    4.3.2 Multiple Conditional Equivalence Statements ............ 98

5 Query-Complexity Amplification 103  
  5.1 QCA for Random Oracles .......................................... 103


## Contents

5.1.1 Query-Complexity Amplifiers ...................................... 103
5.1.2 Number of Queries ................................................... 104
5.2 The Caveats of Plain Iterated Hashing ............................ 110
  5.2.1 Failure as a Query-Complexity Amplifier ................... 110
  5.2.2 An Example of a Vulnerable Application ..................... 113
5.3 The Collision-Free Iteration Amplifier ............................ 114
  5.3.1 The Protocol and the Simulator ................................. 114
  5.3.2 Indistinguishability Proof ...................................... 117
  5.3.3 Globally-Restricted Random Oracles ......................... 120
5.4 Towards Optimality ................................................... 122

II Common Randomness between Two Distrustful Parties 125

6 Common Randomness Amplification ................................. 127
  6.1 Common Randomness Amplifiers ................................... 127
  6.2 Constructing Common Randomness ................................ 130
    6.2.1 Availability Condition ...................................... 131
    6.2.2 Simulatability Against a Malicious Bob ................... 131
    6.2.3 Simulatability Against a Malicious Alice ................ 132
    6.2.4 Combining the Simulatability Conditions ................. 133
  6.3 Impossibility of CRA .............................................. 134
    6.3.1 Perfect Case ................................................ 135
    6.3.2 Statistical Case ............................................ 136

7 Unfair Coin Tossing ................................................... 143
  7.1 Blum’s Coin Tossing Protocol .................................... 143
  7.2 Necessity of an Unfair Coin Tossing Resource ................. 145
  7.3 Constructing UCT .................................................. 147

Bibliography ............................................................ 151
List of Figures

2.1 Non-interactive distinguisher [DS] . . . . . . . . . . 35

3.1 A transformable system $S_T$ . . . . . . . . . . . . . . 51
3.2 Password-based message authentication . . . . . . . . . 56
3.3 Password-based encryption . . . . . . . . . . . . . . . . 62

5.1 A $(\varphi, \varepsilon)$-query-complexity amplifier . . . . . . . . . 105
5.2 Collision-free iteration protocol $\text{amp}_c$ . . . . . . . . . . 115

6.1 Flow of information for a common randomness amplifier . . 131
6.2 A distinguisher internally emulating the converter $\beta$ . . . 133

7.1 Blum’s coin tossing protocol. . . . . . . . . . . . . . . . 146
## List of Systems

2.1 WUF-CMA game $G^{CMA}_{\text{MAC}}$ ........................................ 45  
2.2 IND-CPA system $G^{CPA}_{b}$ (SE) ...................................... 46  

3.1 Core resource $\text{KEY}^r$ .................................................. 52  
3.2 Core resource $\text{SEC}^r$ .................................................. 53  
3.3 Local trigger $\text{LT} (\mathcal{P}, q)$ ....................................... 54  
3.4 Global trigger $\text{GT} (\mathcal{P}, q)$ ....................................... 54  
3.5 Simulator $\sigma_{kd}$ ....................................................... 56  
3.6 Channel $\text{UAUT}^r$ ..................................................... 58  
3.7 Simulator $\sigma_{\text{mac}}$ .................................................... 59  
3.8 Reduction $C^{CMA}_{T,j^*}$ ............................................... 60  
3.9 Distinguisher $D_{\ell}$ ..................................................... 69  
3.10 Converter $\sigma_{\text{KEY}}$ .................................................. 72  
3.11 Converter $\sigma_{\text{SEC}}$ .................................................. 73  
3.12 Simulator $\sigma_{\text{CPA}}$ .................................................. 74  
3.13 Reduction $C^{\text{CPA}}_{j^*}$ ............................................... 75  
3.14 Distinguisher $D'_{\ell}$ .................................................... 82  
3.15 Distinguisher $D_{\ell,q}^{\text{CPA}}$ ............................................ 83  

5.1 Game $Q (\pi, \ell, r)$ to count the number of queries .............. 106  
5.2 Distinguisher $D'$ based on game winner $D$ ....................... 107  
5.3 Distinguisher $D_{\ell,rt}^{\text{iter}}$ ........................................... 111  
5.4 Simulator $\sigma_{\text{amp}}$ .................................................... 117  
5.5 Sampling for $\mathbf{R}$ ..................................................... 119  
5.6 Sampling for $\mathbf{S}^{-1}$ .................................................. 119  
5.7 Distinguisher $D_{\pi,x,k}^{\text{QCA}}$ interacting with $\mathbf{S}$ .................. 123  

6.1 Common reference string resource $\text{CRS} (\mathcal{P}_V)$ ............ 128  
6.2 Common randomness resource $\text{CR} (\mathcal{P}_V)$ .................... 128
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Bit commitment resource COM with abort</td>
<td>144</td>
</tr>
<tr>
<td>7.2</td>
<td>Converter bet</td>
<td>145</td>
</tr>
<tr>
<td>7.3</td>
<td>Converter flip</td>
<td>145</td>
</tr>
<tr>
<td>7.4</td>
<td>Unfair coin tossing resource $\text{UCT}^a$</td>
<td>147</td>
</tr>
<tr>
<td>7.5</td>
<td>Simulator $\sigma_{\text{flip}}$</td>
<td>148</td>
</tr>
<tr>
<td>7.6</td>
<td>Simulator $\sigma_{\text{bet}}$</td>
<td>148</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Randomness in Cryptography

Cryptography was classically devoted to the task of confidential communication, typically in a military context, where messages needed to remain secret to avert untrustworthy or coerced messengers. Although designing encryption schemes was at the time more an art than a science, some classical ciphers already used the idea of sharing beforehand a secret random value to later provide (some form of) confidentiality to sent messages. This is for example the case in the Scytale cipher (known since the war between Sparta and Athens in the 5th century B.C. \cite{Yas02}), where sender and receiver have each a rod of the same diameter and encryption is performed by writing down a message on a strip of parchment wound around the rod.

However, the importance of randomness was not revealed until the seminal work of Shannon \cite{Sha49} who put cryptography on firm mathematical foundations and formally proved that (secret) randomness is necessary in order to achieve confidentiality. Since then, cryptography evolved into a broad scientific field which is not only concerned with confidentiality, but also with authentication, privacy, anonymity, interactive proof systems, digital cash, and more generally performing any kind of computation among distrustful parties.

Despite these tremendous changes, randomness remains at the heart of cryptography and is a core ingredient in all aforementioned applications. Randomness is needed for key generation in both symmetric and asymmetric cryptography, as well as to randomize encryption or to blind digital signatures. Randomness is also needed for the generation of nonces.
which are for example used in key establishment protocols to avoid replay
attacks, or for the generation of challenges in zero-knowledge protocols and
in authentication protocols. Randomness is needed as well to toss a coin
in order to resolve disputes or to play a game like poker over the internet.
There are several different types of randomness, which can be classified
according to the number of parties involved, which of these parties could
potentially be dishonest, whether or not secrecy is required, whether or
not the outcome can be dictated by a single party, etc.

**Constructing randomness resources.** Randomness primitives can be
viewed as resources, where a resource corresponds to a system with an
interface to every party in consideration, e.g., Alice, Bob, and Eve in a
typical communication setting, and with a specific behavior formalizing
the expected guarantees in security definitions. Following the paradigm
of constructive cryptography introduced by Maurer and Renner [MR11,
Mau11], we understand a cryptographic protocol or scheme as a way to
construct, in a well-defined sense, a certain desired resource from one or
more assumed resources. For example, the well-known one-time pad can
be seen as constructing a secure channel from a shared secret key and an
authenticated channel.

This constructive way of stating security definitions comes with a
natural notion of composition. Denoting the statement that a protocol \( \pi \)
constructs the desired resource \( S \) from the assumed resource \( R \) as \( R \xrightarrow{\pi} S \),
any two such construction steps that “syntactically” match can be composed:
If we consider another protocol \( \rho \) that assumes the resource \( S \) and constructs
a resource \( T \), the composition theorem immediately implies that

\[
R \xrightarrow{\pi} S \land S \xrightarrow{\rho} T \implies R \xrightarrow{\rho \circ \pi} T,
\]

where \( \rho \circ \pi \) denotes the composed protocol.

The composability of constructions allows for modularity in designing
new protocols and also in assessing the security of existing schemes. The
composition theorem implies that the security of a complex protocol, which
can be broken down into a sequence of constructions, can be shown by
simply providing a proof for each intermediate construction step, a much
simpler object than the overall combined protocol and which could also be
reused as a building block in other constructions.

**Outline of contributions.** The general landscape of the thesis is the
space of cryptographic constructions involving randomness resources, as
an assumed or as a desired resource.
1.1 Randomness in Cryptography

Password-based cryptography. The main and first part of this thesis focuses on constructions assuming a resource providing cryptographic keys derived from hashing passwords, i.e., randomness which is secret but generally far from uniform. Part I is based on the results presented in publications [DGMT14, DGMT15, DGMT16] and overall give a modular analysis of the rationale behind password-based cryptographic protocols a la PKCS #5.

Chapter 3 develops a new and general technique for modeling resources that can be gradually weakened by an adversary, but still maintains some security guarantees afterwards. This technique is then applied to assess the security of password-based encryption and password-based message authentication. The results of Chapter 3 yield a characterization of when password-derived keys can be used in a composable simulation-based security framework for the task of secure communication. To the best of our knowledge, this work represents the first composable treatment of (non-interactive) password-based encryption and message authentication.

However, passwords are often poorly chosen, falling within a set small enough to be exhaustible. To impede such brute-force attacks, Chapter 5 uses the formalism and tools associated with query-restricted systems developed in Chapter 4 to define the notion of a query-complexity amplifier, a protocol which increases the computational complexity of evaluating a hash function. Chapter 5 also gives an explicit query-complexity amplifier, called collision-free iteration, that provably and generically amplifies the query-complexity by a factor $c$, for any fixed parameter $c$.

Common randomness. The second part of this thesis focuses on construction where the desired resource provides common randomness. The randomness provided does not need to be secret, but it is crucial for fairness that it cannot be influenced by any party. Results of Part II are based on the contents of [DM12, DM13].

Chapter 6 considers the task of extending a given amount of (not necessarily uniform) common randomness by exchanging messages between two distrustful parties and shows that such a task is information-theoretically impossible. That is, any protocol aiming to extend common randomness suffers from a large distinguishing advantage or the entropy gained by executing such a protocol is negligible.

Chapter 7 considers the classical coin tossing resource, a resource which provides the same uniform bit to both parties, and introduces the notion of an unfair coin tossing resource by relaxing guarantees such as fairness and non-influenceability that an ideal coin toss would provide. The presence
of this non-ideal behavior is necessary in order to understand what coin tossing protocols really achieve when a majority of players is dishonest.

We discuss in the next subsections the details of each of these contributions and how they relate to other results in the literature.

1.2 Password-Based Cryptography

1.2.1 Per-Session Security

Human-memorable passwords represent one of the most widely deployed security mechanisms in practice. They are used to authenticate human users in order to grant them access to various resources such as their computer accounts, encrypted files, web services, and many more. Despite well-known problems associated with this mechanism, its practicality and simplicity from the users’ perspective is the main cause of its persisting prevalence. As an example, more than 90% of Google users employ passwords as the only authentication mechanism for accessing their accounts [PTAI15]. Acknowledging this situation, it is extremely important that security engineers, including designers of cryptographic protocols, have a precise understanding of the security guarantees that passwords provide for multiple sessions (where one session corresponds to one password; this is often referred to as the multi-user setting).

There has been significant effort in formalizing the use of passwords, but the standard provable-security approach in cryptography, focusing on a single session, falls short of modeling the expected guarantees. The main reason for this is that passwords, in contrast to cryptographic keys, can be guessed by the attacker with a probability that can hardly be considered negligible in the analysis. This is because they are chosen by the users, and therefore typically do not contain sufficient entropy. When inferring the security guarantees for multiple sessions via the standard hybrid argument, these non-negligible terms from the analyses of the individual sessions accumulate.

To obtain practically relevant statements about systems that allow for many sessions with passwords, we cannot resign on all security guarantees as soon as any password is guessed. Ideally, one would instead hope that as long as not all passwords were broken, the sessions with passwords that are still safe from the attacker enjoy a non-reduced degree of security. This simple yet important observation has been emphasized before, most notably in the work of Bellare et al. [BRT12] on multi-instance security, which we discuss in greater detail below.
From a more general perspective, a setting with multiple sessions relying on passwords can be seen as an instance of a scenario where the considered resource (e.g., a webmail server) can be gradually weakened by the adversary (e.g., by guessing the passwords in some of the sessions), while it is still expected to provide some security guarantees (e.g., for the other sessions) after such weakening.

Contributions
Chapter 3 develops a technique for modeling resources that are used in protocols or applications and can be gradually weakened (we call this “downgrading”). Later, this technique is applied to password-based cryptography and to analyze the security of schemes that use password-derived keys.

Downgradable resources. As our first contribution, we provide in Section 3.1 a natural and intuitive formalization of settings where a considered resource can be potentially downgraded by the actions of an attacker, but still maintains some security guarantees afterwards. While there are many possible ways to analyze such settings, our formalization allows for the natural decoupling of the descriptions of (1) the resource’s behavior at various “levels” of the downgrade; and (2) the mechanism that controls how the system is currently downgraded (as a response to the actions of the attacker). We believe that this modularity allows for simpler analyses of a wide range of resources that can be seen in this way, we discuss the concrete case of password-based cryptography below. The technique is, however, more general, and may also find applications in other scenarios where guarantees may degrade gradually, such as the failure of (some) computational assumptions.

Applications to password-based cryptography. As our second contribution, we apply this modeling approach to several settings that involve multiple sessions using cryptographic keys derived from hashing passwords. The potential downgrading that we consider here corresponds to guessing the passwords in some of the sessions.

Idealizing the hash function as a random oracle, a natural expectation for any such setting is that one obtains a per-session guarantee, i.e., that as long as the attacker does not guess a password in a particular session, the security guarantees provided in this session remain identical to the case where a perfect key is used (i.e., chosen uniformly at random from a large key space). In particular, the security guarantees of one session are
not influenced by other sessions, such as by other users’ poor choice of a password.

We show that this intuitive view is not generally correct. Below we explain the reason of this breakdown (which is a variant of the commitment problem that occurs in adaptive attacks on public-key encryption), and by giving a series of results we draw a map of settings that do/do not succumb to this problem:

1. **Password-based MACs.** We show in Section 3.3 that if password-derived keys are used by a MAC to authenticate insecure channels, a per-session message authentication is achieved.

2. **Single-session PBE.** For password-based encryption (PBE), obtaining a composable statement (i.e., in a simulation-based framework) is much more delicate even in a single-session case. The reason for this is that, roughly speaking, the simulator in the ideal world is expected to produce a simulated ciphertext upon every encryption and without any knowledge of the actual plaintext. However, if the distinguisher later guesses the underlying password (and hence can derive the encryption key), it can easily decrypt the simulated ciphertext and compare the result to the (known) plaintext. But the simulated ciphertext essentially committed the simulator to a message (or a small subset of the message space), so the check will fail with overwhelming probability. Nonetheless, we show in Section 3.4.1 that in the single-session setting designing a simulator, while non-trivial, is possible.

3. **Multi-session PBE.** In line with our motivation, the desired result would be to obtain per-session confidentiality, an analogue of the above single-session statement for the setting with multiple sessions. Surprisingly, as our next contribution, we show in Section 3.4.2 that lifting this positive result to the multi-session setting is unachievable. Roughly speaking, any construction of $r$ secure channels from $r$ authenticated channels and the corresponding $r$ password-derived keys will suffer from a simulation problem analogous to the single-session case described above. However, this time we formally prove that it cannot be overcome. This also solves an open problem posed in the full version of [BRT12].

4. **Multi-session PBE with local assumptions.** To side-step the above impossibility statement, our next result considers the setting of password-based encryption under an additional assumption that the
number of adversarial password guesses in each of the sessions is a pri-
or known, a premise that is also present in [BRT12]. This assumption
seems implausible in general, in fact we show in Section 3.4.4 that
the salting technique often used in the context of password hashing
does not satisfy it; instead, as we also show, salting (only) guarantees
a global upper bound. (Yet, there may be specific settings in which
the validity of the per-session bounds can be argued.) Section 3.4.3
shows, however, that the assumption of local bounds is sufficient to
overcome the commitment problem and prove that the intuitively
expected guarantees described above are indeed achieved. This result
can be seen as a stronger (composable) analogue of the positive result
from [BRT12].

5. **PBE scheme from PKCS #5.** Finally, we observe in Section 3.4.5
that the arguments underlying the above impossibility result in item 3
can also be applied to the password-based encryption as standardized
in PKCS #5 [Kal00].

**Related Work**

**Multi-instance security.** At a very high level, the notion of multi-
instance security proposed in [BRT12] aims at ensuring that, in a setting
where the security of each single session cannot be guaranteed, the amount
of work needed for breaking many sessions cannot be amortized, i.e., it
grows (linearly) with the number of sessions considered.

We believe that this approach, while bringing to light a problem of great
practical relevance, suffers from certain shortcomings that we illustrate on
the example of password-based cryptography. By focusing only on the num-
ber of sessions that can be broken, multi-instance security *cannot* capture
the intuition that sessions protected by strong passwords should be less
vulnerable than sessions protected by weak passwords. Indeed, even if the
results of [BRT12] were to be generalized to deal with arbitrary password
distributions (the results of [BRT12] are phrased for passwords which are
independently and identically distributed), the resulting guarantees would
be in the form of a global upper bound on the number of sessions that
can be broken and would not give any guarantee for a specific session. In
particular, they would not guarantee that a session whose password was
not guessed is secure, independently of whether other sessions were compro-
mised. Finally, [BRT12] only investigates PBE under the assumption that
the number of local password guesses for each of the sessions is a priori
known, which limits the applicability of the results in realistic scenarios
where only a global bound on the attacker’s computational power may be known or assumed.

**Password-based security.** There is an extensive body of work investigating password-based security mechanisms from various perspectives. On the empirical side, the weaknesses of passwords in practice were studied e.g. in [O’G03].

For password-derived keys, most provable-security works focused on the single-session setting, analyzing ways to augment the key-derivation process to slow down offline brute-force password-guessing attacks. Techniques to achieve this include salting (which was introduced in a scenario with multiple users but without a provable-security analysis) [Kal00], iteration [MT79, DGMT15], and hashing with moderately hard-to-compute functions [Per09, AS15, CBS16]. However, the security analyses of those works have a different aim from ours as none of them considers the multi-session scenario. A notable, already mentioned exception is [BRT12] which studied key derivation functions proposed in PKCS #5 [Kal00] and did focus on security in a setting with multiple users.

A key-recovery security definition for password-based encryption was given in [AW05], but here also only single-session security was considered.

Finally, another line of work aims at realizing password-authenticated key exchange (PAKE) protocols [BPR00, GL03, CHK+05] that prevent the possibility of offline password-guessing attacks and result in keys that can then safely be used for encryption or authentication. The protocols are, however, intrinsically interactive and cannot be used in non-interactive password-based settings such as ours.

**The commitment problem.** A simulation problem similar to the one that we described in the context of PBE has already been observed in various other contexts such as public-key encryption with adaptive corruptions [Nie02], functional encryption [BSW11, BO13, MM15] and identity-based encryption [HMM15].

**Multi-user security.** An orthogonal notion called *multi-user security* was studied in [BBM00]. However, its aims are different: the adversary wins if she manages to break a single session, possibly exploiting the presence of other sessions. In contrast, we want to provide guarantees even after some sessions are broken.
1.2 Password-Based Cryptography

1.2.2 Query-Complexity Amplification

Moderately Hard Hashing

Hash functions are one of the most basic and widely used building blocks in practically deployed cryptographic protocols. Their use in different contexts puts diverse requirements on their properties. There is a vast body of literature exploring various desirable properties of cryptographic hash functions such as collision resistance, (second-) preimage resistance, indifferentiability from a random oracle, and several others.

A seemingly orthogonal property of a hash function is its efficiency — quantified by the amount of computational resources that are required to evaluate it. Naturally, the typical design goal is to provide hash functions that are as efficient as possible, while still maintaining the desired security requirements mentioned above. As a result of the long-term design effort with this motivation, the currently standardized and used cryptographic hash functions such as SHA-1, SHA-2 [SHA12] and SHA-3 [SHA14] are extremely efficient: for example, a software implementation of SHA-2 can process data at (very roughly) about 100 MB/s on a typical PC.

However, in several application scenarios the efficiency of the hash function actually has serious security implications, and these motivate design efforts going in the opposite direction. Namely, sometimes hash functions are used to perform computation by the honest parties that would need to be repeated on a significantly higher scale by an adversary trying to compromise the security of the system. One example of such a setting is any non-interactive password-based scheme where the hash function is used to, say, derive a key from this password as described in Section 3.2. Here, increasing the complexity of the hash-function evaluation, while slightly increasing the computational burden for the honest user, also significantly increases the cost of a brute-force and password-guessing (dictionary) attack. Another setting that could benefit from an adjustable complexity of a hash function is a proof of work [DN93] where a legitimate protocol participant shows that he performed a certain amount of computation. This concept was proposed, among other uses, as a countermeasure against denial-of-service attacks or junk mail. Similar ideas are used in the now widely used Bitcoin system [Nak08] and other cryptocurrencies basing their security on proofs of work.

The common denominator of all the settings mentioned above is that it would be desirable to employ hash functions that are, loosely speaking, moderately hard to compute [Nao03]. While the occasional evaluation of such a function by an honest user needs to still remain feasible, at the same
time the scaling resulting from a brute-force attack must be prohibitive for any adversary.

**Complexity amplification.** Since designing new cryptographic hash functions from scratch is a long and intricate process (e.g., the SHA-3 competition spanned over almost 5 years), to answer the above-described demand it would be preferable to give generic schemes that would instead turn an existing hash function $h$ into a new function $H$ with moderately increased evaluation complexity. A natural first candidate for such a scheme is the simple $c$-iteration (or plain iteration), i.e., letting

$$H(\cdot) := h^c(\cdot) := h(...h(\cdot)...),$$

for some integer $c > 1$.

Indeed, many password-hashing schemes are based on some form of iteration. Historically, the earliest implementations of crypt(3) used several iterations of (a variant of) the block cipher DES to hash users’ passwords on Unix systems [MT79], the more recent bcrypt [PM99] iterates the block cipher Blowfish instead. Iteration is also used in the password-based key derivation function PBKDF2 standardized in PKCS #5 [Kal00] and recommended by NIST [TBBC10].

However, when it comes to assessing the security of any such generic scheme for increasing evaluation complexity (for example to justify the choice of plain iteration), it turns out that merely defining the security requirement formally is a surprisingly subtle task. This is especially true if one asks for a composable definition that then allows every scheme secure under this definition to be plugged into any possible application, so that proving a scheme secure according to this single definition immediately implies that it can be used in, e.g., key derivation, proofs of work, or other applications. One of the main contributions of Chapter 5 will be to give such a composable definition by modeling the underlying hash function $h$ as a random oracle and exploiting the well-established notion of indifferentiability (which can be seen as a special case of constructive cryptography [MR11, App. D]). Before inspecting it in greater detail, let us first mention a surprising observation about plain iteration.

**The caveats of plain iteration.** Dodis et al. [DRST12] studied the structural differences between a random oracle and its second iterate: more precisely, they investigated the indifferentiability of the 2-iteration of a
1.2 Password-Based Cryptography

random oracle from a plain random oracle. Interestingly, they showed that such indifferentiability does hold, but only with poor parameters. Namely (and very roughly), any simulator in this indifferentiability statement, if asked \( n \) queries during the distinguishing experiment, would itself have to issue a large number of queries \( \Omega(\ell n) \) to the underlying random oracle in order to succeed in simulation, where \( \ell \) denotes (an upper bound on) the number of honest queries. (We show in Section 5.2 that the result extends to higher-order iterates.) On a high level, this large number of simulator queries means that if one uses the \( c \)-iterate of a hash function in some application, then the concrete security statement obtained through the composition theorem of indifferentiability is weaker than intuitively expected. Therefore, any strong security guarantee could only be obtained through an ad-hoc security analysis depending on the particular scenario considered, as done by Bellare et al. [BRT12].

Hash-then-sign example. To illustrate this last point, let us recall an example of Dodis et al. [DRST12]. In the hash-then-sign paradigm, a signature scheme \( SS\) signing \( n \)-bit messages and a hash function \( h : \{0,1\}^* \to \{0,1\}^n \) are combined into a signature scheme \( SS\) for arbitrary length messages by signing the hash \( h(m) \) of the message instead of the message \( m \) itself. Forging a signature for the extended scheme \( SS\) requires either to find a collision for the hash function \( h \) or to find a forgery for the original fixed-length signature scheme \( SS\). If the hash function \( h \) is modelled as a random oracle, then its second iterate \( h^2 \) is indifferentiable from \( h \) [DRST12, Thm. 2], and the composition theorem of indifferentiability [MRH04, RSS11] implies that the security of \( SS\) can be reduced to that of \( SS\). However, such a reductionist argument, which is standard in any composable cryptographic framework such as indifferentiability, consists of obtaining an adversary against \( SS\) from an adversary against \( SS\) which additionally performs the job of the simulator given in the indifferentiability statement. Due to the blow-up in simulator queries mentioned above, this concretely means that one relates an adversary trying to forge a signature for \( SS\) with at most \( \ell \) signing queries and \( n \) random oracle queries, to an adversary trying to forge a signature for \( SS\) also with \( \ell \) signing queries, but with \( \ell \cdot n \) random oracle queries. Thus, although \( SS\) is secure as long as the collision probability \( (\ell + n)^2 / 2^n \) is sufficiently small (assuming that the original length-restricted signature scheme \( SS\) is secure within \( \ell \) queries), the security of \( SS\) derived through composition depends instead on the much higher collision probability \( (\ell \cdot n)^2 / 2^n \), representing a quadratic
decrease of security.

Contributions

Chapter 5 develops a new formal framework for treating the amplification of the evaluation complexity for random oracles (which are often used to model hash functions in practical scenarios). We first develop a security definition that tightly captures how well a given scheme increases the computational burden for an adversary in evaluating the function. Our definition is naturally composable and hence guarantees the desired universal applicability of any scheme meeting it. Secondly, guided by the observations of Dodis et al. [DRST12] about the second iterate, we show that plain iteration, regardless of the number of iterations employed, fails to achieve the amplification of the hash-function complexity in the above sense. In response, we develop a modification of the plain-iteration scheme, called collision-free iteration, that does provably and generically achieve the desired amplification.

Composable security for hash-complexity amplification. Using the random oracle model (ROM) of Bellare and Rogaway [BR93], we model hash functions as random oracles. A random oracle can be viewed as a resource that is available to all parties in a given setting, and allows each of them to evaluate the oracle by querying it—this corresponds to the party internally computing the output of the hash function. A restriction on the computational resources of the adversary hence naturally translates to a restriction on the number of queries it is allowed to ask the random oracle. In a typical security proof in the ROM, one establishes that the scheme in question is secure unless the ROM-adversary performs a huge number of queries to the random oracle. This then suggests that the adversary against the real implementation has to evaluate the hash function on a prohibitive number of inputs. Following this intuition, we model the increase in evaluation complexity of a hash function by a decrease in the number of queries that the adversary is allowed to issue to the random oracle (before its computational resources are exhausted).

As a starting point, Section 5.1 makes explicit the number of queries that such an oracle allows to each party: for two integers $L$ and $R$, a random oracle that allows up to $L$ queries at the left (honest user’s) interface and up to $R$ queries at the right (adversary’s) interface formalizes the guarantee that the honest user has sufficient resources to evaluate the hash function (at least) $L$ times, whereas the resources of the adversary are bounded to (at most) $R$ evaluations. Naturally, a desirable guarantee for the honest
user is that the number $L$ is large enough to execute higher-level protocols, whereas the number $R$ must be small enough to prevent the adversary from attacking those protocols with significant probability. The goal of a protocol for the amplification of query complexity is hence to reduce the number $R$, while at the same time not affecting the number $L$ more than necessary.

The goal of a query-complexity amplifier (QCA) scheme is then to construct, from a random oracle that allows the adversary to do some number $R$ of queries, a random oracle that allows the adversary only a smaller number $r < R$ of queries. Intuitively, such a construction means that an adversary with the same computational resources can evaluate the random oracle less often, which will generally reduce his success in attacking higher-level protocols. Turned around, Section 5.1.2 shows that this also implies that each evaluation of a query-complexity amplifier scheme requires an unavoidable minimum amount of queries to the random oracle.

**Query-restricted systems.** Chapter 5 heavily relies upon formalism and results of independent interest introduced in Chapter 4. In particular, the notion of parameterized construction statements introduced in Section 4.1 will be central to our definition of hash-complexity amplification and we believe that this formalism will be useful also in many other settings, such as secure communication as discussed in [Tac14]. Section 4.3 shows how conditional equivalence—a central tool in deriving indistinguishability proofs introduced by Maurer [Mau02]—can also be used to derive families of constructions parameterized by the number of queries that can be made to the resources. The results in this section will lay the groundwork for the parameterized constructions shown in Sections 5.3.2 and 5.3.3. As a result of independent interest, Section 4.2 provides a separation result between conditional equivalence and the optimality of non-adaptive strategies.

**A scheme for hash-complexity amplification.** The second contribution of Chapter 5 is to present an explicit and simple scheme, called collision-free iteration, that provably achieves query-complexity amplification in the sense of the new definition discussed above.

One would naturally expect that the $c$-iteration of a random oracle for some $c \geq 2$ would lead to a reduction of adversary queries from $R$ to $R/c$, at the cost of simultaneously reducing the honest party’s queries from $L$ to $L/c$. However, we show in Section 5.2 that $c$-iteration, much like the second iterate studied by Dodis et al. [DRST12], suffers from the blow-up
in the number of simulator queries and therefore falls short of achieving this goal.

We show in Section 5.3 that modifying the c-iterate of a random oracle by a proper encoding of the queries will indeed lead to the desired (and expected) result. The high-level idea is to make sure that each query will access a distinct part of the random oracle and hence the “shifted chains” of queries that caused problems for the plain iteration will not occur. In greater detail, collision-free iteration works almost like the plain iteration, but each query to the underlying function $h(\cdot)$ during the computation of $H(x)$ is prefixed by a prefix-free encoding $\lfloor x \rfloor$ of the original query $x$, as well as the sequence number within the iterative process. Formally, we define $W_0(x)$ to be the empty string and

$$W_j(x) := h(\lfloor x \rfloor \|| j \|| W_{j-1}(x)) \quad \text{for all } j \in \{1, \ldots, c\},$$

where $\lfloor \cdot \rfloor$ and $\langle \cdot \rangle$ denote a prefix-free encoding and an injective encoding of an integer over $\lceil \log c \rceil$ bits, respectively. Finally, we simply let $H(x) := W_c(x)$. We prove in Section 5.3.2 that this construction reduces the number of adversary queries from $R$ to $R/c$, at the cost of simultaneously reducing the honest party’s queries from $L$ to $L/c$. Section 5.3.3 shows that the same collision-free iteration protocol can also reduce the number of adversarial queries that can be made to globally restricted random oracles, which are for example the result of the salting technique as described in Section 3.4.4.

Towards proving optimality. In Section 5.4 we study whether this simultaneous reduction of the honest-party queries is inherent to any query-complexity amplification scheme. Based on the observation that the adversary can always choose to evaluate the honest scheme, we can show that our construction, which reduces the adversary’s queries exactly as much as the honest party’s queries, is optimal with respect to a natural, albeit restricted, class of simulators.

We aimed for simplicity in the design of our construction and did not tailor it to minimize query lengths. In particular, extending the length of each subquery by the length of $\lfloor x \rfloor$ is most likely not necessary. We leave the question of improving the lengths of the honest-user queries open for future work.
1.3 Common Randomness

1.3.1 Common Randomness Amplification

Playing any probabilistic game over the Internet requires some randomness shared among the players. This common randomness is necessary to emulate what could have had happened if the players were physically present: shuffle cards, throw a dice, etc. The common randomness used does not need to be secret, but it is crucial for fairness that it cannot be influenced by any party. Common randomness is also central to many cryptographic applications, such as to realize non-interactive zero-knowledge proofs, e.g., [BFM88, BSMP91, FLS90], or to achieve composable multi-party computation, e.g., [Can01, CLOS02, Lin04].

Chapter 6 focuses on the general problem of common randomness amplification between two distrustful parties. That is, two distrustful parties knowing an initial common random value and having access to an ideal communication channel to exchange messages wish to agree on a new common random value which has higher Shannon entropy than what they initially shared.

Related work. Hofheinz, Müller-Quade and Unruh [HMQU06] studied the problem of coin toss extension, which can be seen as an important subproblem of common randomness amplification,\(^1\) where both the assumed and desired randomness are uniform. Whether or not coin toss extension is possible depends on the type of security: stand-alone or universal composability (formalized in the Universal Composability framework of Canetti [Can01]), and on the desired level of security: computational, statistical, or perfect. The characterization given by Hofheinz et al. [HMQU06] is nearly complete, excepted in the case of statistical composable security, where their impossibility result only holds for protocols which exchange a polynomial number of messages. The question whether inefficient protocols could actually achieve coin toss extension was answered negatively in very recent work by Seiler and Maurer [SM16].

Contributions. The main contribution of Chapter 6 is to prove that the task of common randomness amplification in a composable security framework is information-theoretically impossible, not only for uniform distribution as in [HMQU06] but for any distribution. Our proof is different

\(^1\)This is with respect to the common randomness resource introduced in System 6.2, which contrary to a standard common reference string resource (shown in System 6.1), allows a party to decide when the other one shall receive the random sample.
than that of [HMQU06] and uses mostly classical information-theoretic inequalities. Section 6.3.1 shows that no protocol between two distrustful parties can perfectly increase the Shannon entropy of a given distribution by exchanging any finite amount of messages. When the level of security is relaxed to statistical, Section 6.3.2 shows that common randomness amplification remains impossible, even though our impossibility result suffers from the same caveat as in [HMQU06], i.e., it only holds with respect to protocols which exchange a polynomial number of messages.

We believe that overcoming this last restriction could be done by generalizing the arguments of [SM16] to arbitrary distributions and is left as an opened task.

1.3.2 Unfair Coin Tossing

A two-party ideal coin tossing resource delivers the same uniform random bit to both parties. They are in particular guaranteed that the bit they received is *fair*, the first party receives a bit if and only if the second party receives the same bit, and *not influenced*, the bit a party receives is uniformly random no matter what the other party does. Phrased as a construction, the goal of coin tossing protocols is of course to construct such an ideal coin tossing resource. However, since Cleve’s impossibility result [Cle86], it is known that such an attempt is doomed to fail in the setting of two distrustful parties (more generally any setting where a majority of players is dishonest). In Chapter 7, we ask the following simple question. Since coin tossing protocols cannot construct an ideal coin tossing resource, what do they actually construct?

**Related Work.** Blum [Blu81] gave the first coin tossing protocol under the assumption that one-way functions exist. His motivation was to allow two distrustful parties to emulate a coin toss over the telephone instead of having to be physically present to toss the coin. However, the security of Blum’s protocol critically relies on the ability for the honest party *not* to output anything in case the other party aborts. Fairer coin tossing protocols, where an honest party also outputs a bit when the protocol is aborted, are obviously more desirable. Unfortunately, a celebrated result from Cleve [Cle86] shows that such fairer protocols are inherently biased. That is, a dishonest party can, simply by aborting prematurely the protocol, bias the output of the honest party by as much as \( \Omega(\frac{1}{p}) \), where \( p \) is the number of rounds of the protocol. This impossibility result has usually been tackled in two different manners. Either by restricting the security
1.3 Common Randomness

notion in allowing the honest parties not to output anything in case the protocol is aborted, like in Blum’s protocol. Or, if one still wants some security guarantees after the protocol was aborted, to relax the metric, i.e., the distinguishing advantage, between the ideal coin tossing functionality and what the protocol achieves. The latter approach was initiated by Katz [Kat07] and lead to the notion of an optimally fair coin toss [MNS09, BOO10].

**Contributions.** Unfortunately, security proofs in the second approach are not composable because of the huge relaxation of the metric ($1/q(p)$ for some fixed polynomial $q$ instead of negligible) which could be critical for basic functionalities like coin tossing. Instead, Chapter 7 proposes a very natural approach which consists of making explicit the exact “ideal” resource constructed by the aforementioned coin tossing protocols. Of course, the constructed “ideal” resource will contain some form of non-ideal behavior (fairness and non-influenceability will have to be relaxed) and will be weaker than an ideal coin toss. The goal is not to introduce weaker resources per se, but simply to be able to state exactly what a protocol achieves. We introduce the notion of an unfair coin tossing resource in Section 7.2 and proves in Section 7.3 that Blum’s protocol can be phrased as perfectly constructing such a resource.
Chapter 2

Preliminaries

This chapter introduces notation and concepts used throughout this thesis. Section 2.1 presents some basic notational convention, while Section 2.2 recalls a few key facts about information theory. The main parts of this chapter are Sections 2.3 to 2.5, which are mostly devoted to the abstract and constructive cryptography framework of Maurer and Renner [MR11, Mau11], on which all our security statements are based upon. Finally, Section 2.6 presents some recurrent symmetric cryptographic primitives.

2.1 Basic Notation

Sets and tuples. We denote sets by calligraphic letters or capital Greek letters, e.g., $S$ or $\Sigma$. A tuple $(s_1, \ldots, s_k)$ of $k$ values is denoted $s^k$ and the set of all $k$-tuples of elements of $S$ is denoted $S^k$. For a finite set $S$, the number of its elements is denoted $|S|$. The power set of a set $S$, i.e., the set of all subsets of $S$, is denoted by $2^S$. The disjoint union of $r$ sets $S_1, \ldots, S_r$ is denoted by $\bigcup_{j=1}^r S_j$, where $\bigcup_{j=1}^r S_j := \bigcup_{j=1}^r \{j\} \times S_j$.

A function $f$ with domain $\mathcal{X}$ and codomain $\mathcal{Y}$ is denoted $f : \mathcal{X} \to \mathcal{Y}$ and $\mathcal{Y}^\mathcal{X}$ denotes the set of all such functions. A family of objects over some set $S$ indexed by a set $\mathcal{I}$ is a function $f : \mathcal{I} \to S$ that we denote by $\{s_i\}_{i \in \mathcal{I}}$, where $s_i := f(i)$ for all $i \in \mathcal{I}$. When $\mathcal{I} = \{1, \ldots, k\}$, we will alternatively consider the family $\{s_i\}_{i \in \{1,\ldots,k\}}$ as a tuple $s^k$.

We denote the sets of natural integers and real numbers by $\mathbb{N}$ and $\mathbb{R}$, respectively, while $\mathbb{R}^+$ denotes the set of all non-negative reals. The set of bit strings of finite length is denoted $\{0,1\}^*$ and the empty bit string is denoted $\bot$, whereas $\Diamond$ is used as an error symbol.
**Pseudo-code.** We describe the input/output behavior of systems by using standard pseudo-code notation that should cause no confusion. Deterministic assignments are denoted by \( := \), while probabilistic ones are indicated by an arrow \( \leftarrow \). We write \( x_1, \ldots, x_r \leftarrow S \) to denote that the values \( x_1, \ldots, x_r \) are selected independently and uniformly at random in the set \( S \). A block of pseudo-code starting with "on input \( x \) at \( i \)" describes the actions a system does when queried on input \( x \) at interface \( i \) (see later Section 2.4). Possible actions notably include querying other systems, and the notation "\( z := \text{result of querying} \ y \) at \( i \)" is used to capture that the system being described outputs \( y \) as a query at interface \( i \) and stores the resulting response as \( z \).

**Probability theory.** Throughout this thesis, we consider only discrete random experiments. Recall that a discrete random experiment consists of a pair \( \mathcal{E} := (\Omega, P) \), where \( \Omega \) is the discrete set of elementary events and \( P : \Omega \to [0, 1] \) is a probability distribution mapping each elementary event \( \omega \in \Omega \) to its probability \( P(\omega) \) and such that \( \sum_{\omega \in \Omega} P(\omega) = 1 \). We will sometimes write \( P^\mathcal{E} \) to emphasize the random experiment under consideration. The probability of an event \( A \subseteq \Omega \) is denoted by \( P(A) := \sum_{\omega \in A} P(\omega) \). The conditional probability of \( A \) given another event \( B \subseteq \Omega \) such that \( P(B) > 0 \) is given by \( P(A \mid B) := \frac{P(A \cap B)}{P(B)} \).

A random variable \( X \) over a set \( \mathcal{X} \) is a function \( X : \Omega \to \mathcal{X} \) and induces a probability distribution \( P_X : \mathcal{X} \to [0, 1] \), where \( P_X(x) := \sum_{\omega \in \Omega} x(\omega) = x \ P(\omega) \) for all \( x \in \mathcal{X} \). A tuple \( (X_1, \ldots, X_k) \) of \( k \) random variables, where each \( X_j \) is a random variable over \( \mathcal{X} \), is denoted by \( X^k \) and can be seen as a single random variable over \( \mathcal{X}^k \). The conditional probability distribution of a random variable \( Y \) over \( \mathcal{Y} \) given another random variable \( X \) over \( \mathcal{X} \) is a function \( P_{Y \mid X} : \mathcal{Y} \times \mathcal{X} \to [0, 1] \) defined as \( P_{Y \mid X}(y, x) := \frac{P_{Y \mid X}(y, x)}{P_X(x)} \) for all \( y \in \mathcal{Y} \) and \( x \in \mathcal{X} \) with \( P_X(x) > 0 \).

Frequently, we will define the distribution of a random variable first, while the actual random experiment is only implicitly defined. In particular, \( X \leftarrow \mathcal{X} \) denotes a random variable \( X \) which is uniformly distributed over the set \( \mathcal{X} \). We denote by \( p_{\text{coll}}(k, n) \) the probability that \( k \) independent random variables uniformly distributed over a set of size \( n > k \) collide, i.e., the \( k \) outcomes are not all distinct. It is well-known that

\[
p_{\text{coll}}(k, n) = 1 - \prod_{j=1}^{k-1} \left( 1 - \frac{j}{n} \right) \leq \frac{k^2}{2n}.
\]
2.2 Information Theoretic Measures

Information theoretic measures, such as Shannon entropy or mutual information, will play a key role in Part II of this thesis. We briefly recall here usual definitions and results about these quantities which can be found in standard information theory literature such as [CK11, CT06].

2.2.1 Entropy and Mutual Information

Given a probability distribution \( P \) over \( \Omega \), the entropy \( H(P) \) is a measure of the uncertainty contained in the distribution \( P \), where

\[
H(P) := \sum_{\omega \in \Omega} P(\omega) \cdot \log_2 \left( \frac{1}{P(\omega)} \right),
\]

with the convention that \( 0 \cdot \log_2 \left( \frac{1}{0} \right) = 0 \). For a random variable \( X \) distributed according to \( P_X \), we define \( H(X) := H(P_X) \). Recall that \( 0 \leq H(X) \leq \log_2 |\mathcal{X}| \), where \( H(X) = 0 \) if \( X \) is a constant (i.e., \( P_X(x) = 1 \) for some \( x \in \mathcal{X} \)) and \( H(X) = \log_2 |\mathcal{X}| \) if \( X \) is instead uniformly distributed over \( \mathcal{X} \). The binary entropy function \( h(p) \) corresponds to the entropy of a binary random variable which is 1 with probability \( p \), where

\[
h(p) := p \cdot \log_2 \left( \frac{1}{p} \right) + (1-p) \cdot \log_2 \left( \frac{1}{1-p} \right),
\]

for all \( p \in [0,1] \).

Given two random variables \( X \) and \( Y \) distributed according to some joint distribution \( P_{XY} \) over the set \( \mathcal{X} \times \mathcal{Y} \), the joint entropy \( H(XY) \) of \( X \) and \( Y \) is defined as \( H(P_{XY}) \). Given an event \( A \), the entropy \( H(Y | A) \) of \( Y \) conditioned on \( A \) is defined as \( H\left( P_{Y|A} \right) \). Then, the conditional entropy \( H(Y | X) \) corresponds to the entropy of the distribution \( P_{Y|X=x} \) averaged over all \( x \in \mathcal{X} \), i.e.,

\[
H(Y | X) := \sum_{x \in \mathcal{X}} P_X(x) \cdot H(Y | X = x).
\]

It can be shown that joint and conditional entropy satisfy the relation

\[
H(XY) = H(X) + H(Y | X),
\]

and by definition joint entropy is symmetric in the sense that \( H(XY) = H(YX) \).
The \textit{mutual information} \( I(X; Y) \) between two random variables \( X \) and \( Y \) measures the amount of information that \( Y \) contains about \( X \) and can be defined as
\[
I(X; Y) := H(Y) - H(Y | X),
\]
which from the equation above can also be rewritten as \( H(X) + H(Y) - H(XY) \). Thus, mutual information is symmetric in the following sense
\[
I(X; Y) = I(Y; X)
\]
and it can be shown that \textit{conditioning reduces entropy}, i.e., \( H(Y | X) \leq H(Y) \), which implies that \( I(X; Y) \geq 0 \). The \textit{conditional mutual information} \( I(X; Y | Z) \) of random variables \( X \) and \( Y \) given \( Z \) is defined by
\[
I(X; Y | Z) := H(Y | Z) - H(Y | XZ),
\]
where \( I(X; Y | Z) \geq 0 \) due to the fact that conditioning reduced entropy.

Mutual information and conditional mutual information are such that
\[
I(X; YZ) = I(X; Y) + I(X; Z | Y);
\]
and more generally it can be shown that given \( r + 1 \) random variables \( X_1, \ldots, Y_r \) and \( Y \), entropy and mutual information satisfy the following \textit{chain rules}
\[
H(X_1, \ldots, X_r) = \sum_{j \in \{1, \ldots, r\}} H(X_j | X_1, \ldots, X_{j-1}),
\]
\[
I(X_1, \ldots, X_r; Y) = \sum_{j \in \{1, \ldots, r\}} I(X_j; Y | X_1, \ldots, X_{j-1}).
\]

A superscript will be added to the previous information-theoretic quantities whenever the distribution according to which the random variables are drawn needs to be emphasized, such as in \( H^P(X) \), \( I^P(X; Y) \), and \( I^P(X; Y | Z) \).

We will often use Fano’s inequality [Fan61] to upper bound the conditional entropy of two random variables. The presentation of this classical result follows that of [CK11, Lem. 3.8].

\textbf{Lemma 2.1} (Fano’s inequality). \textit{For two random variables} \( X \) and \( Y \) \textit{with the same range} \( \mathcal{Y} \),
\[
H(Y | X) \leq P(Y \neq X) \cdot \log_2(|\mathcal{Y}| - 1) + h(P(Y \neq X)).
\]
2.2 Information Theoretic Measures

2.2.2 Statistical Distance

Given two probability distributions $P$ and $Q$ over the same set $\Omega$, we denote by $d(P, Q)$ the statistical distance (also known as the total variation distance) between the distributions $P$ and $Q$, where

$$d(P, Q) := \frac{1}{2} \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)| .$$

We also write $d(X, Y) := d(P_X, P_Y)$ for any two random variables $X$ and $Y$ with the same range.

A simple result that we will often use implicitly is that the statistical distance of two joint distributions cannot be smaller than that of their marginals.

**Lemma 2.2.** Let $(X, Y)$ and $(U, V)$ be two pairs of random variables over the same set $\mathcal{X} \times \mathcal{Y}$ distributed according to $P_{XY}$ and $P_{UV}$, respectively. Then,

$$\max \{d(P_X, P_U), d(P_Y, P_V)\} \leq d(P_{XY}, P_{UV}) .$$

**Proof.** The proof simply follows from the triangle inequality. More precisely,

$$d(P_X, P_U) = \frac{1}{2} \sum_{x \in \mathcal{X}} |P_X(x) - P_U(u)|$$

$$= \frac{1}{2} \sum_{x \in \mathcal{X}} \left| \sum_{y \in \mathcal{Y}} P_{XY}(x, y) - P_{UV}(x, y) \right|$$

$$\leq \frac{1}{2} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} |P_{XY}(x, y) - P_{UV}(x, y)|$$

$$= d(P_{XY}, P_{UV}),$$

and similarly for $d(P_Y, P_V) \leq d(P_{XY}, P_{UV})$.

The next simple lemma proves the intuitive result that if two random variables are often equal then their statistical distance must be small.

**Lemma 2.3.** Consider an arbitrary triple of random variables $(X, Y, W)$ over some set $\mathcal{X} \times \mathcal{W} \times \mathcal{W}$ distributed according to a joint distribution $P_{XYW}$. Then,

$$d(P_{XY}, P_{XW}) \leq P(Y \neq W),$$

where $P(Y \neq W)$ denotes the probability that $Y$ is different from $W$ when drawn according to $P_{XYW}$. 

Proof. Let us first compare $P_{XY}(x,y)$ to $P_{XW}(x,y)$, for some $x \in \mathcal{X}$ and $y \in \mathcal{W}$. Recall that $P_{XY}(x,y) = \sum_{y' \in \mathcal{W}} P_{XYW}(x,y,y')$ and similarly $P_{XW}(x,y) = \sum_{y' \in \mathcal{W}} P_{XYW}(x,y',y)$. Thus,

$$P_{XY}(x,y) - P_{XW}(x,y) = \sum_{y' \neq y} P_{XYW}(x,y,y') - P_{XYW}(x,y',y).$$

Using the triangle inequality and the previous equation we can then upper bound the statistical distance as follows

$$d(P_{XY}, P_{XW}) = \frac{1}{2} \sum_{(x,y) \in \mathcal{X} \times \mathcal{W}} |P_{XY}(x,y) - P_{XW}(x,y)| \leq \frac{1}{2} \sum_{(x,y) \in \mathcal{X} \times \mathcal{W}} \sum_{y' \neq y} P_{XYW}(x,y,y') + P_{XYW}(x,y',y) = P(Y \neq W),$$

where the last step follows since $P(Y \neq W) = \sum_{y \in \mathcal{W}} \sum_{y' \neq y} P_{YW}(y,y')$ and trivially $P(Y \neq W) = P(W \neq Y)$. 

Relation to entropy. The following lemma, whose proof can be found in Sason’s work [Sas13, Th. 3], shows that the entropy difference between two distributions reduces as the distributions become statistically closer.

Lemma 2.4. For any two probability distributions $P$ and $Q$ over the same set $\Omega$,

$$|H(P) - H(Q)| \leq d(P, Q) \cdot \log_2(|\Omega| - 1) + h(d(P, Q)).$$

The bound obtained in Lemma 2.4 improves on the standard bound obtained via Pinsker’s inequality [CK11] and is also tight, i.e., there exist distributions $P$ and $Q$ for which the upper bound is actually an equality.

Bias. For a binary random variable $X$, we will sometimes not only be interested in its distance from the uniform distribution over $\{0, 1\}$, which is $|1 - 2 \cdot P_X(1)|$, but also in knowing whether this deviation is towards $0$ or $1$. We denote by $b(X) \in [-1, 1]$ the bias of the bit $X$, where

$$b(X) := 1 - 2 \cdot P_X(1).$$

Note that a bias of $-1$ (respectively, $+1$) corresponds to the constant bit $1$ (respectively, $0$).
2.3 Cryptographic Algebra

To formulate our results we use the language of abstract systems introduced by Maurer and Renner in [MR11, Mau11]. Following the “top-down” approach presented in [MR11], we start in Section 2.3.1 by giving a self-contained introduction to the language of [MR11] for modelling system interactions at a very abstract level. Later in Section 2.4, we refine this view by formalizing the input-output behavior of systems, based on the concept of random systems introduced by Maurer [Mau02]. This last level of abstraction will be sufficient to state all our results. Finally, the three concrete settings considered in this thesis will be detailed in Section 2.5.

2.3.1 Abstract Systems

At the highest level of abstraction, a system is an object with interfaces that are used to interact with other systems. Two systems can be composed by connecting one interface of each system and the resulting object is again a system. Overall, a cryptographic algebra is formed by two types of systems: resources and converters, and by operations between them that we describe below.

Resources. Resources are systems with a finite set \( I \) of interfaces. As a notational convention, we generally use upper-case bold-face letters, such as \( R \) or \( S \) for generic resources, and upper-case sans-serif letters for more specific resources, such as \( \text{KEY} \) for a shared secret key resource or \( \text{AUT} \) for an authenticated channel resource. Most of the resources in this thesis will only have two or three interfaces. Resources with interface set \( I \) are called \( I \)-resources and we denote by \( \Phi \) the set of \( I \)-resources considered.

Multiple \( I \)-resources \( R_1, \ldots, R_r \) can be composed in parallel, denoted \([R_1, \ldots, R_r]\), for some integer \( r \). The resulting system is again an \( I \)-resource, where each interface \( i \in I \) of the composed system \([R_1, \ldots, R_r]\) allows access to the \( i \)-interface of each system \( R_j \), for all \( j \in \{1, \ldots, r\} \).

Converters. Converters are systems with exactly two interfaces: one inside interface that we denote by \( \text{in} \) and one outside interface that we denote by \( \text{out} \). Converters are denoted by lower-case Greek letters (e.g., \( \alpha, \sigma \)) or by sans-serif fonts (e.g., \( \text{enc}, \text{dec} \)). We denote the set of converters by \( \Sigma \).

Converters can be attached to an interface of a resource in order to modify the behavior of the resource at this interface. Attaching the inside interface \( \text{in} \) of a converter \( \alpha \) to the interface \( i \in I \) of an \( I \)-resource \( R \) is denoted by \( \alpha^i R \). The composed system \( \alpha^i R \) is an \( I \)-resource, where its
interface $i$ corresponds to the outside interface out of the attached converter $\alpha$, while other interfaces remain unaffected. In particular, attaching two converters $\alpha$ and $\beta$ at two different interfaces $i$ and $j$ of a same $\mathcal{I}$-resource $R$ commutes in the following sense: $\alpha^i (\beta^j R) = \beta^i (\alpha^i R)$. Attaching no converter to the interface $i$ of an $\mathcal{I}$-resource $R$ is defined as attaching a special identity converter that we denote by $1$ such that $1^i R = R$, for all interfaces $i \in \mathcal{I}$ and all $\mathcal{I}$-resources $R$.

Converters can also be composed sequentially and in parallel. The sequential composition of two converters $\alpha$ and $\beta$ is denoted by $\alpha \circ \beta$ and the resulting system is a converter defined such as $\alpha \circ (\beta \circ \gamma)^i R := \alpha^i (\beta^i R)$, for all interfaces $i \in \mathcal{I}$ and all $\mathcal{I}$-resources $R$. Note that this operation is associative and we will therefore drop the parentheses from now onwards.

The parallel composition of multiple converters $\alpha_1, \ldots, \alpha_r$ is denoted by $\langle \alpha_1, \ldots, \alpha_r \rangle$ and the resulting system is again a converter defined such as $\langle \alpha_1, \ldots, \alpha_r \rangle^i [R_1, \ldots, R_r] := [\alpha_1^i R_1, \ldots, \alpha_r^i R_r]$, for all interfaces $i \in \mathcal{I}$ and all $\mathcal{I}$-resources $R_1, \ldots, R_r$.

The above operations can easily be generalized to deal with families of converters. Given a family of converters $\alpha = \{\alpha_i\}_{i \in \mathcal{I}}$, we denote by $\alpha R$ the resource obtained by attaching the converter $\alpha_i$ at the interface $i$ of $R$ for all $i \in \mathcal{I}$. The sequential composition of two families of converters $\alpha = \{\alpha_i\}_{i \in \mathcal{I}}$ and $\beta = \{\beta_i\}_{i \in \mathcal{I}}$ is naturally defined as their element-wise composition, i.e., $\alpha \circ \beta := \{\alpha_i \circ \beta_i\}_{i \in \mathcal{I}}$. The family of identity converters is denoted by $1$, where $1 := \{1\}_{i \in \mathcal{I}}$.

Similarly, we extend the notion of parallel composition for $r$ converters to the case of families of converters as follows. Given $r$ families of converters $\alpha_1, \ldots, \alpha_r$, where $\alpha_j := \{\alpha_{j,i}\}_{i \in \mathcal{I}}$ for all $j \in \{1, \ldots, r\}$, we denote by $\langle \alpha_1, \ldots, \alpha_r \rangle$ the family of converters where its element indexed by $i$ is $\langle \alpha_1, \ldots, \alpha_r \rangle^i$, for all $i \in \mathcal{I}$.

### 2.3.2 Pseudo-Metric

We also consider a function $d : \Phi \times \Phi \to \mathbb{R}^+$ to measure the similarities between two $\mathcal{I}$-resources. We will assume that $d$ is a pseudo-metric, i.e., for any three $\mathcal{I}$-resources $R$, $S$ and $T$ the function $d$ is such that

1. $d(R, R) = 0$
2. $d(R, S) = d(S, R)$ (symmetry)

For any three converters $\alpha$, $\beta$ and $\gamma$, any $\mathcal{I}$-resource $R$ and any interface $i \in \mathcal{I}$, we have $(\alpha \circ (\beta \circ \gamma))^i R = ((\alpha \circ \beta) \circ \gamma)^i R$. 

2.3 Cryptographic Algebra

3. \( d(R, T) \leq d(R, S) + d(S, T) \) (triangle inequality)

Note that in contrast to a proper metric, the fact that \( d(R, S) = 0 \) does not necessarily imply that \( R = S \).

Typically, the pseudo-metric used in cryptography is defined as the distinguishing advantage of a particular distinguisher. Different distinguishers induce different pseudo-metrics and we will therefore consider a set of pseudo-metrics \( \mathcal{D} \) over the set \( \Phi \) of resources. We postpone the definition of distinguishers to Section 2.4.1 since at this higher level of abstraction only the properties of a pseudo-metric are needed.

**Non-expanding properties.** We now define two important properties in order to characterize how the operations of the cryptographic algebra defined in Section 2.3.1 are handled by the set \( \mathcal{D} \) of pseudo-metrics. These properties will be particularly relevant when we argue about the composition of the construction notion in Section 2.3.3 and are similar to the notion of compatibility for a class of distinguishers defined by Maurer and Renner [MR11].

Given a pseudo-metric \( d \) over the set \( \Phi \) of \( \mathcal{I} \)-resources, we consider two types of pseudo-metrics induced by the aforementioned operations of the cryptographic algebra:

1. Attaching a family of converters \( \alpha := \{\alpha_i\}_{i \in \mathcal{I}} \) induces a new pseudo-metric \( d^\alpha \) defined as

   \[
   d^\alpha(R, S) := d(\alpha R, \alpha S),
   \]

   for all resources \( R, S \in \Phi \).

2. Attaching in parallel several resources \( T_1, \ldots, T_{k-1}, T_{k+1}, \ldots, T_r \in \Phi \), for some integers \( k \in \{1, \ldots, r\} \) and \( r \), induces a new pseudo-metric \( d[T_1^{k-1}, \ldots, T_{k+1}] \) defined as

   \[
   d[T_1^{k-1}, \ldots, T_{k+1}](R, S) := d([T_1^{k-1}, R, T_{k+1}] , [T_1^{k-1}, S, T_{k+1}]),
   \]

   for all resources \( R, S \in \Phi \) and where \( T_m \) is used as a shorthand for \( T_m, T_{m+1}, \ldots, T_n \).

Then, a set \( \mathcal{D} \) of pseudo-metrics is said to be \( \Sigma^\mathcal{I} \)-non-expanding if \( d^\alpha \in \mathcal{D} \), for all pseudo-metrics \( d \in \mathcal{D} \) and all families of converters \( \alpha := \{\alpha_i\}_{i \in \mathcal{I}} \). Similarly, a set \( \mathcal{D} \) of pseudo-metrics is said to be \( \Phi_k \)-non-expanding if \( d[T_1^{k-1}, \ldots, T_{k+1}] \in \mathcal{D} \), for all pseudo-metrics \( d \in \mathcal{D} \) and all resources \( T_1, \ldots, T_{k-1}, T_{k+1}, \ldots, T_r \in \Phi \).
2.3.3 The Construction Notion

The definition of the construction notion depends on which interface can potentially be dishonest. Rather than presenting three construction notions, one for each setting considered in this thesis, we prefer defining the construction notion in full generality and only later specialized it to the three settings considered. Although this approach adds a certain amount of notation, it has the advantage that properties proven for the general construction notion, such as composability, trivially carry over to any specific setting.

To do so, let us consider a subset $\mathcal{P} \subseteq \mathcal{I}$. We denote by $\overline{\mathcal{P}}$ all the interfaces which are not in $\mathcal{P}$, i.e., $\overline{\mathcal{P}} := \mathcal{I} \setminus \mathcal{P}$. Given a family of converters $\alpha := \{\alpha_i\}_{i \in \mathcal{I}}$, we denote by $\alpha_{\mathcal{P}}$ the family of converters whose element indexed by $i \in \mathcal{I}$ is the original converter $\alpha_i$ if $i \in \mathcal{P}$, or the neutral converter $1$ if instead $i \notin \mathcal{P}$.

**Definition 2.1.** Given a collection of adversarial interfaces $\mathcal{A} \subseteq 2^\mathcal{I}$ and a set $\mathcal{D}$ of pseudo-metrics, a family of converters $\pi \in \Sigma^\mathcal{I}$ constructs an $\mathcal{I}$-resource $\mathcal{S}$ from another $\mathcal{I}$-resource $\mathcal{R}$ relative to a family of converters $\sigma \in \Sigma^\mathcal{I}$ and within $\varepsilon := \{\varepsilon_P : \mathcal{D} \to \mathbb{R}^+\}_{P \in \mathcal{A}}$, denoted $\mathcal{R}_{\{{\pi\sigma},\varepsilon\}_{A,D}} \mathcal{S}$, if

$$d(\pi_{\overline{\mathcal{P}}} \mathcal{R}, \sigma_{\mathcal{P}} \mathcal{S}) \leq \varepsilon_P(d),$$

for all set of adversarial interfaces $P \in \mathcal{A}$ and all pseudo-metrics $d \in \mathcal{D}$.

The family of converters $\pi$ (respectively, $\sigma$) in the previous definition will be referred to as the *protocol* (respectively, *simulator*). The resource $\mathcal{R}$ will be referred to as the *assumed* or *real* resource, while the combined resource $\pi_{\overline{\mathcal{P}}} \mathcal{R}$ will be designated as the *real world*. Similarly, $\mathcal{S}$ will be referred to as the *desired* or *ideal* resource, while $\sigma_{\mathcal{P}} \mathcal{S}$ will be designated as the *ideal world*.

Intuitively, a protocol is said to be secure according to Definition 2.1 if whatever adversaries (identified by a set of adversarial interfaces) could do in the real world, they could also do it in the ideal world by incorporating the work of the simulator. Since the desired resource is secure by definition, and both real and ideal worlds are required to be “close” from each other, it means that these adversarial actions in the real world are “harmless” and the protocol is therefore secure.

Note that for $\mathcal{A} = 2^\mathcal{I}$, the construction notion defined above corresponds to the one introduced by Maurer and Renner in [MR11], where all subsets of interfaces are considered as being potentially dishonest.
The next theorem shows that if the set $\mathcal{D}$ of pseudo-metrics used is non-expanding then the construction notion defined above composes. The proof mostly follows that of Maurer [Mau11, Th.1] and is included here only for the sake of completeness since the construction notion in Definition 2.1 is slightly more general than the one originally proposed.

**Theorem 2.1.** Consider a collection of adversarial interfaces $\mathcal{A}$, where $\mathcal{A} \subseteq 2^I$, and a set $\mathcal{D}$ of pseudo-metrics.

1. If the set $\mathcal{D}$ of pseudo-metrics is $\Sigma^I$-non-expanding, then the construction notion $\overrightarrow{\mathcal{A}, \mathcal{D}}$ is sequentially composable, i.e.,

$$ R \xrightarrow{\mathcal{A}, \mathcal{D}} S \land S \xrightarrow{\mathcal{A}, \mathcal{D}} T \implies R \xrightarrow{\mathcal{A}, \mathcal{D}} T, $$

where $\eta := \{ \eta_P : \mathcal{D} \to \mathbb{R}^+ \}_{P \in \mathcal{A}}$ and $\eta_P (d) := \varepsilon_P (d^\mathcal{A}) + \delta_P (d^\mathcal{S})$, for all $P \in \mathcal{A}$ and all $d \in \mathcal{D}$.

2. If the set $\mathcal{D}$ of pseudo-metrics is $\Phi^r_k$-non-expanding, then the construction notion $\overrightarrow{\mathcal{A}, \mathcal{D}}$ is composable in parallel, i.e.,

$$ R \xrightarrow{\mathcal{A}, \mathcal{D}} S \implies [T^{k-1}_1, R, T^{r}_{k+1}] \xrightarrow{\mathcal{A}, \mathcal{D}} [T^{k-1}_1, S, T^{r}_{k+1}], $$

where $\eta := \{ \eta_P : \mathcal{D} \to \mathbb{R}^+ \}_{P \in \mathcal{A}}$ and $\eta_P (d) := \varepsilon_P \left( d_{[T^{k-1}_1, \cdot, T^{r}_{k+1}]} \right)$, for all sets $P \in \mathcal{A}$ of adversarial interfaces, all pseudo-metrics $d \in \mathcal{D}$, and all resources $T_1, \ldots, T_{k-1}, T_{k+1}, \ldots, T_r \in \Phi$. The protocol $\rho$ is defined as $\rho := (1^{k-1}, \pi, 1^r_{k+1})$ and similarly for the simulator $\tau := (1^{k-1}, \sigma, 1^r_{k+1})$, where the symbol $T^n_m$ (respectively, $1^n_m$) is used as a shorthand for $T_m, T_{m+1}, \ldots, T_n$ (respectively, $1, 1, \ldots, 1$).

**Proof.** Consider a set $P \in \mathcal{A}$ of adversarial interfaces and a pseudo-metric $d \in \mathcal{D}$. We first prove the sequential composition property of the construction notion and then show that it is also composable in parallel.

**Proof of sequential composition.** We upper bound the distance between the resources $\rho_P \pi_P R$ and $\sigma_P \tau_P T$ via the triangle inequality as follows,

$$ d (\rho_P \pi_P R, \sigma_P \tau_P T) \leq d (\rho_P \pi_P R, \rho_P \sigma_P S) + d (\rho_P \sigma_P S, \sigma_P \tau_P T), $$

where we introduced $\rho_P \sigma_P S$ as an intermediate resource. Note that the same family $\rho_P$ of converters is attached to both resources $\pi_P R$ and $\sigma_P S$.

\footnote{Composition in parallel was referred to as context-insensitivity in [MR11].}
The set $\mathcal{D}$ of pseudo-metrics is assumed to be $\Sigma^f$-non-expanding and thus $d^{\rho \pi} \in \mathcal{D}$. Then, the first assumed construction implies

$$d (\rho \pi \rho \pi R, \rho \pi \sigma \pi S) = d^{\rho \pi} (\pi \rho \pi R, \sigma \pi S) \leq \varepsilon (d^{\rho \pi}) \;.$$ 

The distance between the resources $\rho \pi \sigma \pi S$ and $\sigma \pi \tau \pi T$ can similarly be upper bounded by first noticing that $\rho \pi \sigma \pi S = \sigma \pi \rho \pi S$. Since the set $\mathcal{D}$ of pseudo-metrics is assumed to be $\Sigma^f$-non-expanding, it follows that $d^{\sigma \pi} \in \mathcal{D}$, and as the same family $\sigma \pi$ of converters is attached to both $\rho \pi S$ and $\tau \pi T$, the second assumed construction implies

$$d (\rho \pi \sigma \pi S, \sigma \pi \tau \pi T) = d^{\sigma \pi} (\rho \pi S, \tau \pi T) \leq \delta (d^{\sigma \pi}) \;.$$ 

Combining the previous equations finishes the proof of sequential composition.

**Proof of parallel composition.** Recall that the converter indexed by an interface $i \in \overline{\mathcal{I}}$ in the family $\rho \overline{\mathcal{I}}$ is by definition $\langle 1^k_{i}, \pi_i, 1^r_{i} \rangle$. The definition of parallel composition of converters in Section 2.3 then implies

$$\rho \overline{\mathcal{I}} \left[ T^{k-1}_{1}, R, T^{r}_{k+1} \right] = \left[ T^{k-1}_{1}, \pi \overline{\mathcal{I}} R, T^{r}_{k+1} \right],$$

and a similar argument applies for the ideal resource, i.e.,

$$\tau \pi \overline{\mathcal{I}} \left[ T^{k-1}_{1}, S, T^{r}_{k+1} \right] = \left[ T^{k-1}_{1}, \sigma \pi S, T^{r}_{k+1} \right].$$

The same resources $T^{k}_{1}, \ldots, T^{k}_{k-1}, T^{r}_{k+1}, \ldots, T^{r}_{r}$ are in parallel of both $\pi \overline{\mathcal{I}} R$ and $\sigma \pi S$, and since the set $\mathcal{D}$ of pseudo-metrics is assumed to be $\Phi^r_{k}$-non-expanding it follows that $d^{[T_{1}^{k-1}, \ldots, T_{k+1}^{r}]} \in \mathcal{D}$. The assumed construction then implies

$$d \left( [T^{k-1}_{1}, \pi \overline{\mathcal{I}} R, T^{r}_{k+1}], [T^{k-1}_{1}, \sigma \pi S, T^{r}_{k+1}] \right) \leq \varepsilon (d^{[T^{k-1}_{1}, \ldots, T^{r}_{k+1}]}).$$

Combining the previous equations finishes the proof of parallel composition. 

\[\square\]

**Filtered resources.** It is sometimes desirable to distinguish between actions that are guaranteed to honest parties, from actions that are possible to dishonest parties. To model such situations, we use the notion of a **filtered resource** as described in [Mau11, Def. 18]. Given an $\mathcal{I}$-resource $R$ and a family of converters $\phi := \{ \phi_i \}_{i \in \mathcal{I}}$, the filtered resource $R_\phi$ is an $\mathcal{I}$-resource where for a party connected to interface $i$ of $R_\phi$ interactions through the
2.4 Random Systems

converter $\phi_i$ are guaranteed to be available, while direct interactions with $R$ at interface $i$ are only available to dishonest parties.

The construction notion in Definition 2.1 extends naturally to the case of filtered resources as follows. Given two filtered resources $S_\psi$ and $R_\phi$, we write $R_\phi \xrightarrow{(\pi, \sigma, \epsilon)} A, D S_\psi$ if

$$d(\pi_{\mathcal{F}} \phi_{\mathcal{F}} R, \sigma_{\mathcal{F}} \psi_{\mathcal{F}} S) \leq \epsilon_{\mathcal{P}}(d),$$

for all set of adversarial interfaces $\mathcal{P} \in A$ and all pseudo-metrics $d \in D$. Constructions between filtered resources can also be shown to be composable as in Theorem 2.1.

2.4 Random Systems

We instantiate the general concept of abstract systems given above by considering random systems, a framework which was first introduced in the work of Maurer [Mau02]. Many cryptographic primitives like block ciphers, MAC schemes, random functions, etc., can be described at this lower level of abstraction as a random system where only its input-output behavior is relevant. An $(\mathcal{X}, \mathcal{Y})$-random systems takes inputs $X_1, X_2, \ldots \in \mathcal{X}$ and generates for each input $X_k$ an output $Y_k \in \mathcal{Y}$. In full generality, such an output $Y_k$ depends probabilistically on all the previous inputs $X^k = (X_1, \ldots, X_k)$ as well as all the previous outputs $Y^{k-1} = (Y_1, \ldots, Y_{k-1})$. For an $(\mathcal{X}, \mathcal{Y})$-random system $S$, such a dependency is captured by a (possibly infinite) sequence of functions $p_{Y_k|X^kY^{k-1}}^{S} : \mathcal{Y} \times \mathcal{X}^k \times \mathcal{Y}^{k-1} \rightarrow [0, 1]$ such that for all choices of the arguments $x^k$ and $y^{k-1}$ the sum of the function values over the choices of $y_k$ equals 1, and where the superscript indicates the considered system. Random systems are usually denoted by upper-case boldface letters such as $R$ or $S$. An $(\mathcal{X}, \mathcal{Y})$-random system $S$ considered in isolation does not define a random experiment since the distribution of the inputs to the system $S$ is not defined. For this reason, the function $p_{Y_k|X^kY^{k-1}}^{S}$, which is called a conditional probability distribution, is denoted by a lower-case letter $p$ instead of an upper-case letter $P$, which we use for probability distributions in a fully specified random experiment.

**Definition 2.2** ([Mau02]). An $(\mathcal{X}, \mathcal{Y})$-random system $S$ is a (possibly infinite) sequence of conditional probability distributions $p_{Y_k|X^kY^{k-1}}^{S}$, where $X_k \in \mathcal{X}$ and $Y_k \in \mathcal{Y}_k$, for all $k \geq 1$.

A random system $S$ can alternatively be described by the sequence of
conditional distributions \( p_{Y^k|X^k} \), where

\[
p_{Y^k|X^k} := \prod_{j=1}^{k} p_{Y_j|X_j Y_{j-1}}.
\]

Note that the conditional distribution \( p_{Y^k|X^k} \) contains the conditional distribution \( p_{Y_j|X_j} \) for all \( j < k \) and hence the above description of a system is redundant. The conditional distribution \( p_{Y^k|X^k} \) must satisfy a consistency condition which ensures that \( Y_j \) does not depend on \( X_{j+1}, \ldots, X_k \). Two random systems \( R \) and \( S \) are said to be equivalent, denoted \( R \equiv S \), if they behave identically, i.e., \( p_{Y^k|X^k}^R = p_{Y^k|X^k}^S \), for all \( k \geq 1 \).

2.4.1 Distinguishers

A natural notion of similarity for random systems can be based on the concept of distinguishing. Intuitively, a distinguisher \( D \) can be viewed as a system that connects to a random system \( R \), interacts with this system, and at the end of this random experiment outputs a single bit denoted \( B \). In the case of \((\mathcal{X}, \mathcal{Y})\)-random systems, a distinguisher \( D \) that makes some arbitrary but fixed number \( q \in \mathbb{N} \) of queries corresponds to a finite \((\mathcal{Y}, \mathcal{X})\)-random system which is one query ahead [MPR07], i.e., distributions \( p_{X_k|Y_{k-1} X_{k-1}}^D \) for \( k \in \{1, \ldots, q\} \), and an additional distribution \( p_{B|Y^q X^q}^D \). The distinguisher interacts with an \((\mathcal{X}, \mathcal{Y})\)-random system \( R \) by providing inputs \( X_1, X_2, \ldots \in \mathcal{X} \) to \( R \) and by receiving its corresponding outputs \( Y_1, Y_2, \ldots \in \mathcal{Y} \). The complete interaction of \( D \) and \( R \) defines a random experiment and the probability that the bit \( B \) is 1 in this experiment is written as \( P_{DR} (B = 1) \). For two random systems \( R \) and \( S \), the distinguishing advantage of a distinguisher \( D \) in telling apart \( R \) from \( S \) is then defined as

\[
\Delta^D (R, S) := \left| P_{DR} (B = 1) - P_{DS} (B = 1) \right|.
\]

For a class \( \mathcal{D} \) of distinguishers, we define \( \Delta^\mathcal{D} (R, S) := \sup_{D \in \mathcal{D}} \Delta^D (R, S) \).

We consider two classes of distinguishers using the following notation:

- The class of all distinguishers, in which case we omit the superscript and write \( \Delta (R, S) \).

- The class \( \mathcal{NA} \) of all non-adaptive distinguishers. A distinguisher \( D \) is non-adaptive if it selects all its queries \( X_1, \ldots, X_q \) in advance, independently of the outputs \( Y_1, \ldots, Y_q \), i.e., \( p_{X_k|Y_{k-1} X_{k-1}}^D = p_{X_k|X_{k-1}}^D \) for all \( k \in \{1, \ldots, q\} \).
2.4 Random Systems

2.4.2 Games

A central tool in deriving an indistinguishability proof between two systems is to characterize both systems as being equivalent until a certain condition arises [Mau02, BR06]. Thus, being able to distinguish both systems requires to provoke this condition, and one is then interested in upper-bounding the probability of this event. Interacting with a random system in order to provoke a certain condition is naturally modeled by the notion of a game [Mau02]. A game is a random system which replies to every input $X_k$ by an output $Y_k$ and a bit $A_k$ indicating whether or not the game has been won. This binary output is monotone in the sense that it is initially set to 0 and that, once it has turned to 1 (indicating that the game has been won), it can not turn back to 0. An $(\mathcal{X}, \mathcal{Y} \times \{0, 1\})$-system with such a monotone binary output (MBO) is often indicated by using a system symbol with a hat, such as $\overline{R}$.

For an $(\mathcal{X}, \mathcal{Y} \times \{0, 1\})$-system $\overline{R}$ with an MBO, we consider two particular $(\mathcal{X}, \mathcal{Y})$-systems which are derived from $\overline{R}$, following Maurer et al. [MPR07]:

1. $\overline{R}^*$ is the $(\mathcal{X}, \mathcal{Y}')$-system obtained from $\overline{R}$ by ignoring the MBO, we usually refer to this system as $R$ (i.e., we simply omit the hat);

2. $\overline{R}^\dagger$ is the $(\mathcal{X}, \mathcal{Y} \cup \{\ast\})$-system which masks the $\mathcal{Y}$-output to a dummy symbol $\ast \not\in \mathcal{Y}$ as soon as the MBO turns 1, and in addition, it does not output the MBO itself.\footnote{This definition deviates from the one used by Maurer et al. [MPR07], where the MBO is still output by $\overline{R}^\dagger$. The difference between the definitions is irrelevant because the output is $\ast$ if and only if the MBO is 1.}

We will alternatively refer to an $(\mathcal{X}, \mathcal{Y} \times \{0, 1\})$-random system $\overline{R}$ with an MBO as an $(\mathcal{X}, \mathcal{Y})$-game, in particular if we are interested in the probability with which the MBO can be provoked. More formally, we are then interested in the probability that some distinguisher $D$ provokes the MBO of a game $\overline{R}$ to be 1. As in a distinguishing experiment, the interaction of the distinguisher $D$ with the game $\overline{R}$ defines a random experiment and we denote by $A$ the value of the MBO of $\overline{R}$ after $D$ stops. The probability that the bit $A$ is 1 in this experiment is written as $p^D(\overline{R}) (A = 1)$ and the winning probability of $D$ in the game $\overline{R}$ is defined as

$$\Gamma^D(\overline{R}) := p^D(\overline{R}) (A = 1).$$

Similarly to $\Delta^D$, the supremum of $\Gamma^D(\overline{R})$ over $D$ is denoted $\Gamma^D(\overline{R})$.\footnote{This definition deviates from the one used by Maurer et al. [MPR07], where the MBO is still output by $\overline{R}^\dagger$. The difference between the definitions is irrelevant because the output is $\ast$ if and only if the MBO is 1.}
We describe two techniques due to Maurer [Mau02, Mau13] to upper bound the distinguishing advantage between two random systems based on the notion of game winning: game equivalence and conditional equivalence.

**Game equivalence.** The next definition formalizes what it means for two systems to be equivalent until a certain condition arises: the two induced games behave identically as long as they are not won.

**Definition 2.3 ([Mau13]).** Two \((\mathcal{X}, \mathcal{Y})\)-games \(\mathcal{R}\) and \(\mathcal{S}\) with an MBO \(A_1, A_2, \ldots\) are said to be *game equivalent*, denoted \(\mathcal{R} \equiv^{g} \mathcal{S}\), if

\[
p_{Y^k, A_k = 0 | X^k} = p_{Y^k, A_k = 0 | X^k},
\]

for all \(k\).

The next lemma combines two results from Maurer [Mau02, Mau13]. The first states that if two games are game equivalent, then the probability of winning is the same in both games. The second captures the intuitive statement that if two systems are equivalent until a certain event, corresponding to the two induced games being game equivalent, then distinguishing these systems require to provoke this event, i.e., to win the game.

**Lemma 2.5 ([Mau13, Lem. 1,2]).** Let \(\mathcal{R}\) and \(\mathcal{S}\) be two games such that \(\mathcal{R} \equiv^{g} \mathcal{S}\). Then, for any distinguisher \(D\)

\[
\Delta^D(\mathcal{R}, \mathcal{S}) \leq \Gamma^D(\mathcal{R}) = \Gamma^D(\mathcal{S}).
\]

**Conditional equivalence.** The notion of *conditional equivalence* has been introduced by Maurer [Mau02, Mau13] and is a useful tool in deriving indistinguishability proofs. It can be seen as a stronger requirement than game equivalence and will allow us to obtain a stronger analog of Lemma 2.5.

**Definition 2.4 ([Mau13]).** An \((\mathcal{X}, \mathcal{Y})\)-game \(\mathcal{R}\) with MBO \(A_1, A_2, \ldots\) is said to be *conditionally equivalent* to an \((\mathcal{X}, \mathcal{Y})\)-random system \(S\), denoted \(\mathcal{R} \equiv^{c} S\), if

\[
p_{Y^k | X^k, A_k = 0} = p_{Y^k | X^k},
\]

for all \(k \geq 1\) and for all arguments for which \(p_{Y^k | X^k, A_k = 0}\) is defined.

If a game \(\mathcal{R}\) is conditionally equivalent to a system \(S\), then the distinguishing advantage between the systems \(R\) and \(S\) is upper bounded by
Figure 2.1. The dashed box represents the non-interactive distinguisher $[DS]$ which interacts with a game $\mathcal{R}$ by ignoring its outputs and instead interacts with $S$ to produce the sequence of inputs $X_1, X_2, \ldots$ given to $\mathcal{R}$.

the probability of winning the game $\mathcal{R}$ in a non-adaptive manner, a statement which was first presented by Maurer [Mau02] and was studied more extensively later by Jetchev et al. [JÖS12] and Maurer [Mau13].

\textbf{Lemma 2.6 ([Mau13, Th. 3])}. Let $\mathcal{R}$ be an $(\mathcal{X}, \mathcal{Y})$-game and $S$ be an $(\mathcal{X}, \mathcal{Y})$-random system such that $\mathcal{R} \equiv S$. Then, for any distinguisher $D$

$$\Delta^D(\mathcal{R}, S) \leq \Gamma^D(\mathcal{R}) = \Gamma^{[DS]}(\mathcal{R}),$$

where $[DS]$ denotes the non-adaptive distinguisher which interacts only with the system $S$ and forwards every query made to $S$ to the game $\mathcal{R}$ (see Figure 2.1).

\section{2.4.3 Restricted Systems and Games}

The concept of a blocked system $\mathcal{R}^{-}$, derived from a given system $\mathcal{R}$ with MBO, is particularly useful if $\mathcal{R}$ is in turn derived from some underlying system $\mathcal{R}$ (i.e., $\mathcal{R}^{-} = \mathcal{R}$, where $\mathcal{R}$ is of interest to us) by adding an MBO representing some restriction on $\mathcal{R}$ (e.g., an upper bound on the number of queries than can be made to this system). In this case, the restricted distinguishing advantage of a distinguisher $D$ in distinguishing the two systems with MBO $\mathcal{R}$ and $\mathcal{S}$ is defined as

$$\hat{\Delta}^D(\mathcal{R}, \mathcal{S}) := \Delta^D(\mathcal{R}^{-}, \mathcal{S}^{-}). \tag{2.1}$$

Note that the “standard” distinguishing advantage defined in Section 2.4.1 can be seen as a special case of the restricted distinguishing advantage. This is because any non-restricted system $\mathcal{R}$ can be formalized as a restricted system $\mathcal{R}$ where the added MBO modelling the restriction
is always 0. In this case, \( R = \overline{R}^{-1} \) and given another non-restricted system \( S \) modelled in the same manner we have \( \Delta^D (R, S) = \tilde{\Delta}^D (R, S) \).

The concept of restricting a system via an additional MBO can also be applied to the case of games and game winning. In such a case, we consider a system restricted by some MBO \( A_1, A_2, \ldots \) with an additional MBO \( B_1, B_2, \ldots \) specifying when the game is won. Formally, \( R \) is an \((\mathcal{X}, \mathcal{Y} \times \{0,1\} \times \{0,1\})\)-random system, where the outputs \((Y_k, A_k, B_k)\) are triples and the latter two components are monotone. Then, we can consider the task of winning the restricted game, i.e., provoking the event modeled by the MBO \( B_1, B_2, \ldots \) before violating the restriction modeled by the MBO \( A_1, A_2, \ldots \), as the task of winning the game with the MBO \( C_1, C_2, \ldots \) with \( C_k = C_{k-1} \lor (\neg A_k \land B_k) \). Denoting the system with the single MBO \( C_1, C_2, \ldots \) as \( R < \), we define the restricted game-winning advantage as

\[
\hat{\Gamma}^D (R) := \Gamma^D (R <) .
\]

Similarly to \( \Delta^D \) and \( \tilde{\Delta}^D \), the “standard” game winning advantage \( \Gamma^D \) defined in Section 2.4.2 can be seen as a special case of the restricted game-winning advantage \( \hat{\Gamma}^D \).

Given an \((\mathcal{X}, \mathcal{Y})\)-game \( \overline{R} \) outputting pairs \((Y_j, B_j)\) and an MBO \( A_1, A_2, \ldots \) modeling a restriction which is a function of the inputs, such as the number of queries that can be made to \( \overline{R} \), we denote by \( \overline{R}^{A} \) the system outputting triples \((Y_j, A_j, B_j)\) obtained by inserting the MBO \( A_1, A_2, \ldots \) into the output of \( \overline{R} \).

### 2.4.4 Multiple-Interface Random Systems

Systems that can be accessed by multiple parties can be viewed as systems with multiple interfaces and formalized as random systems by making the interface identifier an explicit part of the input (or output) of the system. Resources and converters described as abstract systems in Section 2.3.1 will be instantiated as such multiple-interface random systems.

**Resources as random systems.** We restrict our considerations to the particular class of resources that only produce an output (from some set \( \mathcal{Y} \)) in response to an input (from \( \mathcal{X} \)) and on the same interface (from \( \mathcal{I} \)) where the input was received, and hence we omit the interface label from the output. Then, such a resource \( R \) takes as input a pair \((I_k, X_k) \in \mathcal{I} \times \mathcal{X} \), where the \( k \)-th query \( X_k \) was input at the \( I_k \)-interface, and produces as output \( Y_k \in \mathcal{Y} \), where it is understood that the response \( Y_k \) of the resource...
\( \mathbb{R} \) is output at the same interface \( I_k \) that the query \( X_k \) was input. In other words, an \( \mathcal{I} \)-resource \( \mathbb{R} \) corresponds (due to our restrictions) to an \( (\mathcal{I} \times \mathcal{X}, \mathcal{Y}) \)-random system and can be described by a sequence of conditional probability distributions \( p^R_{Y_k|I^k \times X^k, Y^k-1}, \ k \geq 1 \). Such an \( \mathcal{I} \)-resource with an additional MBO is then simply an \( (\mathcal{I} \times \mathcal{X}, \mathcal{Y} \times \{0,1\}) \)-random system.

The system \([\mathbb{R}_1, \ldots, \mathbb{R}_r]\) resulting from the parallel composition of several of such \( \mathcal{I} \)-resources \( \mathbb{R}_1, \ldots, \mathbb{R}_r \), where each \( \mathbb{R}_j \) is some \( (\mathcal{I} \times \mathcal{X}_j, \mathcal{Y}_j) \)-random system, allows individual access to each system \( \mathbb{R}_1, \ldots, \) or \( \mathbb{R}_r \). This requires that part of the input to the system \([\mathbb{R}_1, \ldots, \mathbb{R}_r]\) specifies which of the \( r \) systems is to be queried. More formally, \([\mathbb{R}_1, \ldots, \mathbb{R}_r]\) is an \( (\mathcal{I} \times \mathcal{X}, \mathcal{Y}) \)-system, where \( \mathcal{X} \) and \( \mathcal{Y} \) are the disjoint union of the \( r \) input sets \( \mathcal{X}_1, \ldots, \mathcal{X}_r \) and of the \( r \) output sets \( \mathcal{Y}_1, \ldots, \mathcal{Y}_r \), respectively, i.e., \( \mathcal{X} := \bigsqcup_{j=1}^r \mathcal{X}_j \) and \( \mathcal{Y} := \bigsqcup_{j=1}^r \mathcal{Y}_j \). The behavior of the combined system \([\mathbb{R}_1, \ldots, \mathbb{R}_r]\) is defined as follows. On input \((i, (j, x)) \in \mathcal{I} \times \mathcal{X}\), the system \([\mathbb{R}_1, \ldots, \mathbb{R}_r]\) outputs \((j, y) \in \mathcal{Y}\), where \( y \) is the response of the \( j \)th resource \( \mathbb{R}_j \) when queried on \((i, x)\). Note that the system obtained by composing in parallel several \( \mathcal{I} \)-resources is again an \( \mathcal{I} \)-resource. As a shorthand, we will often refer to the interface \( i \) of the resource \( \mathbb{R}_j \) in the combined resource \([\mathbb{R}_1, \ldots, \mathbb{R}_r]\) as the interface \( i_j \).

Since \( (\mathcal{I} \times \mathcal{X}, \mathcal{Y}) \)-games are random systems, they can of course be composed in parallel as defined above. However, note that if each \( \overline{\mathbb{R}}_j \) is an \( (\mathcal{I} \times \mathcal{X}_j, \mathcal{Y}_j) \)-game, then the combined system \([\overline{\mathbb{R}}_1, \ldots, \overline{\mathbb{R}}_r]\) is not necessarily an \( (\mathcal{I} \times \mathcal{X}, \mathcal{Y}) \)-game, where \( \mathcal{X} := \bigsqcup_{j=1}^r \mathcal{X}_j \) and \( \mathcal{Y} := \bigsqcup_{j=1}^r \mathcal{Y}_j \). Indeed, the binary output of \([\overline{\mathbb{R}}_1, \ldots, \overline{\mathbb{R}}_r]\) comes from one of the games \( \overline{\mathbb{R}}_j \), depending on which game was queried, and thus is in general not monotone. In order to obtain an \( (\mathcal{I} \times \mathcal{X}, \mathcal{Y}) \)-game, the binary output of \([\overline{\mathbb{R}}_1, \ldots, \overline{\mathbb{R}}_r]\) is changed to be the disjunction of each MBO output by one of the games \( \overline{\mathbb{R}}_j \) and the resulting game is denoted by \([\overline{\mathbb{R}}_1, \ldots, \overline{\mathbb{R}}_r]\). That is, \([\overline{\mathbb{R}}_1, \ldots, \overline{\mathbb{R}}_r]\) is an \( (\mathcal{I} \times \mathcal{X}, \mathcal{Y} \times \{0,1\}) \)-system, where the binary component is 1 if and only if at least one of the games \( \overline{\mathbb{R}}_j \) is won.

**Converters as random systems.** We consider that a converter is always invoked by queries \( X_1, X_2, \ldots \in \mathcal{X} \) at the outside interface \textout. For each such query, it (adaptively) makes zero or more\(^4\) queries \( X'_1, \ldots, X'_{k_1} \) (resp., \( X'_{k_1+1}, \ldots, X'_{k_2} \) etc.) at the inside interface \textin, i.e., to the \( \mathcal{I} \)-resource whose \( i \)-interface is attached to the \textin-interface of the converter. After hav-

\(^4\)We assume that, for each converter, there is some (constant) upper bound on the number of inside queries it makes per outside query.
ing received the corresponding answers $Y'_1, \ldots, Y'_{k_1}$ (resp., $Y'_{k_1+1}, \ldots, Y'_{k_2}$ etc.), it finally produces an output $Y_1 \in \mathcal{Y}$ (resp., $Y_2$ etc.) at the output interface. As it is always clear at which interface the input to the converter is obtained (it is the same interface where the converter produced the last output), it need not be explicitly specified. Summarizing the above, such a converter can be formalized as an $(\mathcal{X} \cup \mathcal{Y}, \mathcal{Z})$-random system, where $\mathcal{Z} := (\{\text{out}\} \times \mathcal{Y}) \cup (\{\text{in}\} \times \mathcal{X})$. Occasionally, the converter considered will have an additional MBO, in which case it is formalized as an $(\mathcal{X} \cup \mathcal{Y}, \mathcal{Z} \times \{0, 1\})$-random system.

Attaching a converter to the $i$-interface of an $I$-resource, where $i \in I$, results in an $I$-resource that can be described as follows.\(^5\) Inputs to interfaces $i' \neq i$ are processed by the system as before. Whenever an input is given to the $i$-interface of the combined system, the converter is evaluated on this input. If the output of the converter (without the potential MBO) is of the form $(\text{in}, x)$ for some $x \in \mathcal{X}$, the resource is evaluated on $(i, x)$ and provides an output $y \in \mathcal{Y}$ (and possibly an MBO). Then, the converter is evaluated on $y$. This process continues until the output of the converter is of the form $(\text{out}, y')$ for some $y' \in \mathcal{Y}$, and this value $y'$ is then considered the output of the composed system. This process leads to a well-defined random system because the number of inside queries is bounded for each query to the random system. The MBO of the overall system is defined to be the disjunction of the MBOs of the $(I \times \mathcal{X}, \mathcal{Y} \times \{0, 1\})$-random system and the converter.

The system $\langle \alpha_1, \ldots, \alpha_r \rangle$ resulting from the parallel composition of several such converters $\alpha_1, \ldots, \alpha_r$, where each $\alpha_j$ is some $(\mathcal{X}_j \cup \mathcal{Y}_j, \mathcal{Z}_j)$-random system with $\mathcal{Z}_j := (\{\text{out}\} \times \mathcal{Y}_j) \cup (\{\text{in}\} \times \mathcal{X}_j)$, is defined similarly to the parallel composition of resources described in the previous paragraph. More formally, $\langle \alpha_1, \ldots, \alpha_r \rangle$ is an $(\mathcal{X} \cup \mathcal{Y}, \mathcal{Z})$-random system, where $\mathcal{Z} := ((\{\text{out}\} \times \mathcal{Y}) \cup (\{\text{in}\} \times \mathcal{X}))$ with $\mathcal{X} := \bigsqcup_{j=1}^r \mathcal{X}_j$ and $\mathcal{Y} := \bigsqcup_{j=1}^r \mathcal{Y}_j$. On input $(j, z) \in \mathcal{X} \cup \mathcal{Y}$, $\langle \alpha_1, \ldots, \alpha_r \rangle$ outputs $(i, (j, z'))$, where $z'$ is the output at the $i$-interface of the $j$th converter $\alpha_j$ at interface $i \in \{\text{out, in}\}$. Note that the system $\langle \alpha_1, \ldots, \alpha_r \rangle$ is again a converter and together with the notion of parallel composition of $I$-resources defined above we have

$$\langle \alpha_1, \ldots, \alpha_r \rangle^i [R_1, \ldots, R_r] = [\alpha_1^i R_1, \ldots, \alpha_r^i R_r],$$ \hspace{1cm} (2.2)

for all interfaces $i \in I$.

---

\(^5\)The described process can be written as a closed formula to formally obtain a random system.
2.5 Three Settings

Distinguishers and reductions. Distinguishers were defined in Section 2.4.1 for arbitrary random systems and are thus in particular defined for the set $\Phi$ of $I$-resources considered in this thesis which are simply $(I \times \mathcal{X}, \mathcal{Y})$-random systems. It can easily be seen that the distinguishing advantage $\Delta^D$ of a distinguisher $D$ is a pseudo-metric as defined in Section 2.3.2 and the set $\mathcal{D}$ of all distinguishers for $(I \times \mathcal{X}, \mathcal{Y})$-random systems induces therefore a set of pseudo-metrics over the set $\Phi$ of resources. The non-expanding properties described in Section 2.3.2 naturally translate into closure properties for the set $\mathcal{D}$.

We will often make reductions between two distinguishing problems. That is, a distinguisher $D$ trying to tell apart two resources $U$ and $V$ is transformed into a new distinguisher whose task is instead to distinguish the resource $R$ from $S$. Such a reduction is done by exhibiting a reduction system $C$ which translates one setting into the other. More formally, such a reduction system $C$ is a converter (with one inside and one outside interface), where the inside interface of $C$ connects to the merged interfaces of the resource $R$, denoted $CR$, and the outside interface of $C$ can be accessed by a distinguisher. To reduce the task of distinguishing $U$ from $V$ to that of $R$ and $S$, one exhibits a reduction system $C$ such that $U \equiv CR$ and $V \equiv CS$. For all distinguishers $D$ it then follows that $\Delta^D(U, V) = \Delta^D(CR, CS) = \Delta^{DC}(R, S)$, where the last equality comes from the fact that $C$ can also be thought of as being part of the distinguisher.

2.5 Three Settings

All the security statements made in this thesis are an instantiation of the construction notion exposed in Section 2.3.3 and are therefore composable in the sense of Theorem 2.1.

Concretely, the resources and converters considered are random systems as exposed in Section 2.4. Random systems will be in general defined via a high level description (often given in pseudo-code) with the implicit understanding that the corresponding sequence of conditional probability distributions could be derived from this description. The description of a random system will be also sometimes simplified by relaxing the requirement that the output should be at the same interface the input was given, the underlying idea being that the described system could be amended\(^6\), if necessary by “additional” signalisation messages, to fit the formalism described in Section 2.4.

---

\(^6\)As an example consider the case of the authenticated channel $\text{AUT}$ in the (Alice,
The set of pseudo-metrics between resources will be the set of distinguishing advantages induced by all distinguishers as defined in Section 2.4.1; or, if the resources considered are restricted, the set of restricted distinguishing advantages (induced by all distinguishers). Alternatively, since a distinguishing advantage can be cast as a restricted distinguishing advantage as discussed in Section 2.4.3, all constructions could be stated with respect to the restricted distinguishing advantage. However, since only Chapters 4 and 5 will formally use restricted-systems, the former option avoids introducing trivial MBOs (which are always 0) and is therefore preferred. The set $\mathcal{I}$ of interfaces considered, as well as which ones among these are adversarial, will depend on the exact setting considered. In this thesis, we consider three settings that we now introduce together with setting-specific notation.

### 2.5.1 The (Alice, Bob, Eve)-Setting

In this setting, two parties Alice and Bob wish to communicate securely despite the presence of an eavesdropper Eve. In such a setting, resources have therefore three interfaces, which we naturally label by elements of the set $\mathcal{I} := \{A, B, E\}$, for Alice’s, Bob’s and Eve’s interface, respectively. The set $\mathcal{A} \subseteq 2^\mathcal{I}$ of adversarial interfaces is then $\mathcal{A} := \{\emptyset, \{E\}\}$ depending on whether or not the adversary is present.

A family of converters $\{\alpha_i\}_{i \in \{A, B, E\}}$ over $\mathcal{I}$ will be written as a triple $(\alpha_A, \alpha_B, \alpha_E)$. Note that Eve is only introduced as a thought experiment to define security but is not meant as a party participating in a protocol with Alice and Bob. Therefore, although a protocol in such a setting formally corresponds to a triple of converters of the form $(\alpha, \beta, 1)$, we will instead denote the protocol by the pair $(\alpha, \beta)$, where $\alpha$ models Alice’s actions taken at her interface $A$, while $\beta$ models Bob’s actions taken at his interface $B$.

Typically, we do not want to guarantee any actions at interface $E$, but simply make some available if the adversary is present. This is for example the case with authenticated channels where it is not guaranteed that Eve learns the transmitted message, rather she only learns it if she is present (as an adversary). Similarly, a secure channel only leaks the length of Bob, Eve)-setting (see Section 2.5.1) described as follows: upon input $x$ at interface $A$, output $x$ at interfaces $B$ and $E$. The described authenticated channel $\text{AUT}$ is formally not a random system since an input at $A$ yields two outputs: one at $B$ and another at $E$. This issue can be resolved by considering instead a channel in which transmitted messages are “pulled”, i.e., on input $x$ at $A$, this channel would return a dummy message at $A$, while the other parties Bob and Eve could retrieve the message input by Alice only after having input $\text{getmsg}$ at their respective interface. Such technicalities are omitted for the sake of clarity.
2.5 Three Settings

the transmitted message if the adversary is present. The behavior of a resource thus depends on whether or not the adversary is present and such a two-mode resource can be modeled by the use of filters as described in Section 2.3.3. In this setting, since Alice and Bob are always honest, a filter is simply a triple of converters of the form $(1, 1, \phi)$, and as a shorthand we will denote the corresponding filtered resource as $R_\phi$ instead of $R_{(1,1,\phi)}$. We are now ready to state the construction notion in the (Alice, Bob, Eve)-setting.

**Definition 2.5.** Let $\varepsilon_1$ and $\varepsilon_2$ be two functions mapping each distinguisher $D$ to a real number in $[0, 1]$. A two-party protocol $\pi := (\alpha, \beta) \in \Sigma^2$ constructs in the (Alice, Bob, Eve)-setting a filtered resource $S_\psi$ from an assumed filtered resource $R_\phi$ relative to a simulator $\sigma \in \Sigma$ and within $\varepsilon := (\varepsilon_1, \varepsilon_2)$, denoted $R_\phi \xrightarrow[A_{BE}]{}_{\{\pi, \sigma, \varepsilon\}} S_\psi$, if

$$
\begin{align*}
\Delta^D (\alpha^A \beta^B \phi^E R, \psi^E S) &\leq \varepsilon_1 (D) \quad \text{(availability)} \\
\Delta^D (\alpha^A \beta^B R, \sigma^E S) &\leq \varepsilon_2 (D) \quad \text{(security)},
\end{align*}
$$

for all distinguishers $D$.

All the constructions in the (Alice, Bob, Eve)-setting stated in this thesis are such that the availability condition is trivially satisfied and we therefore omit it from now onwards. That is, we write $R_\phi \xrightarrow[A_{BE}]{}_{\{\pi, \varepsilon_2\}} S_\psi$ for $R_\phi \xrightarrow[A_{BE}]{}_{\{\pi, \sigma, (0, \varepsilon_2)\}} S_\psi$ and where $R_\phi$ (respectively, $S_\psi$) is implicitly understood from $R$ (respectively, $S$). The (Alice, Bob, Eve)-setting will be used in Chapter 3.

2.5.2 Indifferentiability

The notion of indifferentiability was introduced by Maurer et al. [MRH04] as a generalization of indistinguishability for settings where some access to the internal state of the considered resources is available publicly, within reach of any potential adversary. Indifferentiability has been widely applied, especially in the context of hash functions [CDMP05, BDPV08] and reductions among idealized primitives [HKT11]. In such a scenario, resources have two interfaces: A so-called “private” interface, modeling interaction with honest users; and a “public” interface, formalizing adversarial access.

We take advantage of the fact that resources in such a setting only have two interfaces to lighten the notation: We will understand the left and the right side of the symbol $R$ as representing the private and the public
interface of the resource $R$, respectively. Hence, attaching a converter $\pi$ to the left (private) interface of a resource $R$ results in a resource $\pi R$ while attaching a converter $\sigma$ to the right (public) interface of a resource $S$ results in a resource $S \sigma$. In such a setting, interfaces will be labelled by an element of the set $\mathcal{I} := \{\text{left}, \text{right}\}$ and the set of adversarial interfaces corresponds to $\mathcal{A} := \{\emptyset, \{\text{right}\}\}$.

Similarly to what was done for the (Alice,Bob, Eve)-setting in Section 2.5.1, we denote a family of converters $\{\alpha_i\}_{i \in \{\text{left}, \text{right}\}}$ by a pair $(\alpha_{\text{left}}, \alpha_{\text{right}})$. Even though a protocol in this setting is formally a pair of converters $(\pi, 1)$, we will refer to the converter $\pi$ as being the actual protocol. The behavior of a resource in this setting also depends on whether or not the adversary is present and will be modeled by filtered resources. In the setting of indifferentiability, a filter is a pair of converters $(1, \perp)$ where the converter $\perp$ blocks all inputs and outputs at the interface it is attached to, i.e., formally $\perp$ replies to any query made to its outside interface out by the dummy value $\bullet$ without making any query at its inside interface in. The filtered resource $R_{(1, \perp)}$ will be denoted by $R_{\perp}$.

**Definition 2.6.** Let $\varepsilon_1$ and $\varepsilon_2$ be two functions mapping each distinguisher $D$ to a real number in $[0,1]$. A protocol $\pi \in \Sigma$ constructs in the indifferentiability setting a filtered resource $S_{\perp}$ from an assumed filtered resource $R_{\perp}$ relative to a simulator $\sigma \in \Sigma$ and within $\varepsilon := (\varepsilon_1, \varepsilon_2)$, denoted $R_{\perp} \xrightarrow{\text{IND}_{S_{\perp}}} S_{\perp}$, if

\[
\begin{align*}
\Delta^D (\pi R_{\perp}, S_{\perp}) & \leq \varepsilon_1 (D) \quad \text{(availability)} \\
\Delta^D (\pi R, S \sigma) & \leq \varepsilon_2 (D) \quad \text{(security)},
\end{align*}
\]

for all distinguishers $D$.

Unless stated otherwise, all constructions in the indifferentiability setting stated in this thesis are such that the availability condition is trivially satisfied and will therefore be omitted. That is, we write $R \xrightarrow{\text{IND}_{S_{\perp}}} S$ for $R_{\perp} \xrightarrow{\text{IND}_{S_{\perp}}} S_{\perp}$. The indifferentiability setting will be used in Chapters 4 and 5.

**Parameterized constructions.** Looking ahead, Chapters 4 and 5 will use constructions that are parameterized by the number of queries that can be made to the resources and the goal is then to show that for each value of the parameter a certain construction holds. Formally, such a restriction
on the number of queries can be modelled as an MBO and the resources obtained correspond to restricted systems which can then be distinguished using the restricted distinguishing advantage as defined in Section 2.4.3. The reason to model these restrictions by MBOs is that they allow to capture a notion of uniformity: the parameterization only affects when a random experiment shall stop but not the input/output behavior of the underlying resources.

2.5.3 Two Distrustful Parties

In this setting, two parties Alice and Bob wish to execute a protocol together, although they do not trust each other, i.e., Alice or Bob, but not both, could be dishonest and try to cheat the other party. Similarly to what was done for the indifferentiability setting in Section 2.5.2, we will understand the left and the right side of the symbol $R$ as representing Alice’s and Bob’s interface of the resource $R$, respectively. Hence, attaching a converter $\alpha$ to the left (Alice’s) interface of a resource $R$ results in a resource $\alpha R$ while attaching a converter $\beta$ to the right (Bob’s) interface of $R$ results in a resource $R \beta$. In such a setting, interfaces will be labelled by an element of the set $\mathcal{I} := \{\text{left}, \text{right}\}$ and the set of adversarial interfaces corresponds to $\mathcal{A} := \{\emptyset, \{\text{left}\}, \{\text{right}\}\}$.

Similarly to what was done for the other settings, we denote a family of converters $\{\alpha_i\}_{i \in \{\text{left}, \text{right}\}}$ by a pair $(\alpha_{\text{left}}, \alpha_{\text{right}})$. The behavior of a resource in this setting also depends on whether or not the adversary is present and will be modeled by filtered resources as described in Section 2.3.3.

**Definition 2.7.** Let $\varepsilon_1$, $\varepsilon_2$ and $\varepsilon_3$ be three functions mapping each distinguisher $D$ to a real number in $[0, 1]$. A two-party protocol $\pi := (\alpha, \beta) \in \Sigma^2$ constructs in the two distrustful parties setting a filtered resource $S_\psi$ with $\psi := (\psi_\ell, \psi_r)$ from an assumed filtered resource $R_\phi$ with $\phi := (\phi_\ell, \phi_r)$ relative to a simulator $\sigma := (\nu, \tau) \in \Sigma^2$ and within $\varepsilon := (\varepsilon_1, \varepsilon_2, \varepsilon_3)$, denoted

$$R_\phi \xrightarrow{\pi, \sigma, \varepsilon}_{\text{TDP}} S_\psi,$$

if

$$\begin{align*}
\Delta^D(\alpha \phi_\ell R \phi_\ell \beta, \psi_\ell S \psi_r) & \leq \varepsilon_1(D) \quad \text{(availability)}, \\
\Delta^D(\alpha \phi_\ell R, \psi_\ell S \tau) & \leq \varepsilon_2(D) \quad \text{(security against Bob)}, \\
\Delta^D(R \phi_\ell \beta, \nu S \psi_r) & \leq \varepsilon_3(D) \quad \text{(security against Alice)},
\end{align*}$$

for all distinguishers $D$.

All constructions in the two distrustful parties setting stated in this thesis are information-theoretic. That is, the functions $\varepsilon_1$, $\varepsilon_2$ and $\varepsilon_3$ are
actually each a constant function mapping every distinguisher to some constant. With a slight abuse of notation, such constant functions will be denoted by the constant they map to. The two distrustful parties setting will be used in Part II of this thesis.

2.6 Symmetric Cryptographic Primitives

2.6.1 The Random Oracle Model

The random oracle model (ROM) consists of idealizing a real-world hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$ as a random function $F$ which on a fresh input $x \in \{0, 1\}^*$ outputs an $n$-bit string $y$ selected uniformly at random in $\{0, 1\}^n$ while repeated queries are answered consistently. The random oracle is formalized as the resource $RO$ which on input $x$ at interface $i$ outputs $F(x)$ at the same interface, where $i \in \{A, B, E\}$ if the (Alice, Bob, Eve)-setting is considered or $i \in \{\text{left}, \text{right}\}$ if instead the indifferentiability setting is considered (see Section 2.5). The random oracle methodology was explicitly introduced in the work of Bellare and Rogaway [BR93] and employed in a vast amount of security proofs such as [FS87, Sch91, BR95, BR96, FOPS01].

However, Canetti et al. [CGH98] have shown that there exist schemes that are secure in the ROM, but become completely insecure once the random oracle is instantiated by any hash function. Therefore, if we instantiate the random oracle by an existing cryptographic hash function $h$, the proof in the ROM can then only be taken as a heuristic argument towards the security of the overall construction, even if so far no practical instantiations of schemes with a security proof in the ROM have been broken.

Nonetheless, if one uses a hash function construction $h^f$ that was proven indifferentiable (see Section 2.5.2) from a random oracle when using an ideal compression function $f$, this excludes any possible attacks exploiting the structure of $h$ and reduces the security of the construction to the security of the underlying compression function $f$ [CDMP05]. This is a much more compact object and is comparatively simpler to analyze. As a consequence, an indifferentiability proof from a random oracle is generally considered an important argument towards the security of a practical hash function design and many of the SHA-3 candidates (including the winner Keccak [NIS15]) enjoy such a proof (see [BDPV08, CN08, DRRS09, DRS09, AMP10]).
2.6 Symmetric Cryptographic Primitives

2.6.2 MAC Schemes

Definition 2.8. A MAC scheme with message space $\mathcal{M} \subseteq \{0, 1\}^*$, key space $\mathcal{K} := \{0, 1\}^n$, and tag space $\mathcal{U} \subseteq \{0, 1\}^*$ is defined as a pair $(\text{tag}, \text{vrf})$, where $\text{tag}$ is a (possibly probabilistic) function taking as input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$ to produce a tag $u \leftarrow \text{tag}(k, m)$, and $\text{vrf}$ is a deterministic function taking as input a key $k \in \mathcal{K}$, a message $m \in \mathcal{M}$ and a tag $u \in \mathcal{U}$ to output a bit $b := \text{vrf}(k, m, u)$ asserting the validity of the input tag $u$.

A MAC scheme is correct if $\text{vrf}(k, m, \text{tag}(k, m)) = 1$, for all keys $k \in \mathcal{K}$ and all messages $m \in \mathcal{M}$.

System 2.1: WUF-CMA game $G_{\text{CMA}}(\text{MAC})$

\[
\begin{align*}
\text{win} &:= 0, k \leftarrow \mathcal{K}, \mathcal{B} := \emptyset \\
\text{on input} \ (\text{tag}, m) &\quad \mathcal{B} := \mathcal{B} \cup \{m\} \\
&\quad \text{output} \ \text{tag}(k, m) \\
\text{on input} \ (\text{vrf}, m, u) &\quad b := \text{vrf}(k, m, u) \\
&\quad \text{win} := \text{win} \lor (b \land (m \not\in \mathcal{B})) \\
&\quad \text{output} \ b
\end{align*}
\]

A MAC scheme $\text{MAC} := (\text{tag}, \text{vrf})$ is considered weakly unforgeable under chosen-message attack (WUF-CMA) if it is computationally infeasible, even when given access to an oracle producing valid tags for chosen messages, to generate a valid tag for a fresh message that was not queried before to the oracle. The associated security game, denoted $G_{\text{CMA}}(\text{MAC})$ is detailed in System 2.1.

2.6.3 Encryption Schemes

Definition 2.9. A symmetric encryption scheme with message space $\mathcal{M} \subseteq \{0, 1\}^*$, key space $\mathcal{K} := \{0, 1\}^n$, and ciphertext space $\mathcal{C} \subseteq \{0, 1\}^*$ is defined as a pair $(\text{enc}, \text{dec})$, where $\text{enc}$ is a (possibly probabilistic) function taking as input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$ to produce a ciphertext $c \leftarrow \text{enc}(k, m)$, and $\text{dec}$ is a deterministic function taking as input a key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$ to output a plaintext $m' := \text{dec}(k, c)$. The output of $\text{dec}$ can also be the error symbol $\bullet$ to indicate an invalid ciphertext.
An encryption scheme is correct if \( \text{dec}(k, \text{enc}(k, m)) = m \), for all keys \( k \in K \) and all messages \( m \in M \).

**System 2.2: IND-CPA system \( G_{\text{CPA}}^b(SE) \)**

\[
\begin{align*}
  &k \leftarrow K \\
  &\text{on input } m_0 \in M \\
  &\quad m_1 \leftarrow \mathcal{M}_{|m_0|} \\
  &\quad c \leftarrow \text{enc}(k, m_b) \\
  &\text{output } c
\end{align*}
\]

To define the security of an encryption scheme we use the “real or random” definition of indistinguishability under chosen-plaintext attack (IND-CPA) as it matches more closely constructive security definitions which involve the comparison of a “real” system with an “ideal” one. Relations to other notions of IND-CPA follow from the work of Bellare et al. [BDJR97]. Thus, an encryption scheme \( SE := (\text{enc}, \text{dec}) \) is said to be IND-CPA-secure if no efficient distinguisher can tell apart the system \( G_{\text{CPA}}^b(SE) \), which encrypts input messages, from the system \( G_{\text{CPA}}^1(SE) \), which encrypts random messages of the same length as the input messages. The system \( G_{\text{CPA}}^b(SE) \) is described in System 2.2, where \( \mathcal{M}_\ell \) stands for messages of length \( \ell \) in \( \mathcal{M} \).
Part I

Password-Based Cryptography in the Random Oracle Model
Chapter 3

Per-Session Security

This chapter provides in Section 3.1 a simple technique to capture security guarantees that gracefully degrade and applies it to formalize the notion of per-session security. We then show in Section 3.3 that password-based message authentication does provide per-session security. In contrast, we show in Section 3.4 that this is in general not the case for password-based encryption (PBE). The analysis of PBE is more involved and we therefore start in Section 3.4.1 with the simpler case of a single session to later show in Section 3.4.2 a general impossibility result for multiple sessions. We show in Section 3.4.3 that this impossibility result can be overcome by considering additional local assumptions (as in [BRT12]) and we justify in Section 3.4.4 why these local assumptions appear unrealistic in general. Finally, we observe in Section 3.4.5 that our impossibility result of Section 3.4.2 also applies to the password-based encryption standard PKCS #5 [Kal00].

This chapter is stated in the (Alice, Bob, Eve)-setting described in Section 2.5.1. Resources in this chapter have therefore an interface $A$ for Alice, an interface $B$ for Bob and an interface $E$ for Eve, and the set of interfaces is thus $I := \{ A, B, E \}$.

3.1 Transformable Systems

In this section, we present our approach to modeling systems that can be gradually transformed, in a way that clearly separates the effects of the transformation from how it can be provoked.
3.1.1 Core Systems and Triggers

As a warm-up example, consider a key obtained by hashing a secret password shared between two users Alice and Bob. Idealizing the hash function as a random oracle, the resulting key is completely random from the perspective of any third party Eve unless she also queried the random oracle on the same input; in other words, unless she correctly guessed the password.

Hence, if we model the key obtained by this process as a resource, we consider two separate parts of it. The first one specifies the behavior of the resource before and after the transformation (a “strong” version gives the key only to Alice and Bob, a “weak” version also gives it to Eve); the second part triggers one of these two versions based on Eve’s actions (providing a password-guessing game for her, triggering the weaker version as soon as she wins).

In general, a transformable system is thus the combination of two random systems: a core and a trigger system. The core system specifies how it behaves as an internal switch value changes, while the trigger system specifies how this switch value can be changed. More formally, a core system $S$ is simply an $(\mathcal{X} \cup S, \mathcal{Y})$-random system, where the set of inputs is partitioned into two sets $\mathcal{X}$ and $S$ with $\mathcal{X} \cap S = \emptyset$. The set $\mathcal{X}$ is the set of “normal” inputs, while $S$ is the set of possible switch values, such as $\{0, 1\}$ in our example above. A trigger system $T$ is a $(T, S)$-random system which outputs a switch value. Elements of $T$ are called trigger values and correspond to password guesses in our example above.

**Definition 3.1.** Let $\mathcal{X}, \mathcal{Y}, S$ and $T$ be four discrete sets such that $\mathcal{X} \cap S = \emptyset$ and $\mathcal{X} \cap T = \emptyset$. An $(\mathcal{X} \cup S, \mathcal{Y})$-random system $S$ and a $(T, S)$-random system $T$ form an $(\mathcal{X} \cup T, \mathcal{Y})$-random system, denoted $S_T$, defined as follows. On input $x \in \mathcal{X}$, the system $S_T$ outputs $y \in \mathcal{Y}$, where $y$ is the output of the system $S$ when queried on the input $x$. On input $t \in T$, the system $S_T$ outputs $y' \in \mathcal{Y}$, where $y'$ is the output of $S$ when queried on the output $s \in S$ of the system $T$ which was queried on the original input $t$ (see Figure 3.1).

The random system $S_T$ will be referred to as a transformable system, the random system $S$ as a core system, and the random system $T$ as a trigger system.

Note that a transformable system is just a particular type of a random system, hence any security definition applying to random systems (e.g. the construction notion of Definition 2.5) also applies to transformable systems.
3.1 Transformable Systems

3.1.1 Transformable Systems

Figure 3.1. A transformable system $S_T$ formed by combining a core system $S$ with a trigger system $T$. “Normal” inputs $x \in \mathcal{X}$ are processed directly by $S$, while trigger values $t \in \mathcal{T}$ go instead first through the system $T$ whose output $s \in S$ is then used as an input to the system $S$.

**Fixed Switches.** Given an $(\mathcal{X} \cup S, \mathcal{Y})$-core system $S$, it will be sometimes convenient to argue about the behavior of $S$ for a particular fixed switch value $s \in S$. To do so, we denote by $S_s$ the $(\mathcal{X}, \mathcal{Y})$-random system obtained by initializing $S$ as follows: the switch value $s$ is initially input to $S$ and its resulting output is discarded. In other words, $S_s$ corresponds to the system $S$ where the value of its switch is fixed from the beginning to $s$ and cannot be changed. In particular, the input space of $S_s$ is only $\mathcal{X}$ and not $\mathcal{X} \cup S$. Given a random variable $S$ over $S$, we denote by $S_S$ the system selected at random in $\{S_s \mid s \in S\}$ according to $S$.

3.1.2 Downgradable Resources

The core systems that we will consider will actually be resources, i.e., random systems with 3 interfaces $A$, $B$ and $E$ for Alice, Bob, and Eve, respectively, where the switch values are controlled via the interface $E$. Formally, we model this interface as being split into two sub-interfaces: $E_N$ (for “normal” inputs/outputs) and $E_S$ (for switch values). Resources obtained by fixing the switch of such core resources to a particular value will no longer have this interface $E_S$. Typically, Eve will not have a direct access to the interface $E_S$ of the core resource, instead she will only be allowed to access a trigger system $T$, which itself produces the switch values. Neither Alice nor Bob have access to $T$. Such a core resource combined with a trigger system will be called a downgradable resource.

We now introduce downgradable key resources and downgradable secure channels, examples of such resources that will be used throughout this chapter. These resources are parameterized (among other) by a fixed number $r$ of sessions. Intuitively speaking, these resources provide a graceful deterioration of security by associating each session with a password and guaranteeing that a session remains secure as long as its password is not guessed, irrespectively of the state of other sessions. We first describe the corresponding core resources and then the trigger systems.
Example 3.1 (Key). The core resource $\text{KEY}^r$ for $r$ sessions takes as switch at interface $E_S$ an $r$-bit string $(s_1, \ldots, s_r)$ which specifies for each session whether it is “broken” ($s_j = 1$) or not ($s_j = 0$). Alice and Bob can retrieve a uniform and independent key for a given session, while Eve can only retrieve it if the session is marked as “broken”. The resource $\text{KEY}^r$ is formalized$^1$ in System 3.1.

System 3.1: Core resource $\text{KEY}^r$

\[
\begin{align*}
    s_j &:= 0 \text{ and } k_j \leftarrow \{0, 1\}^n, \text{ for all } j \in \{1, \ldots, r\} \\
    \text{on input } (j, \text{getkey}) &\text{ at } i \in \{A, B\} \\
    &\text{output } (j, k_j) \text{ at } i \\
    \text{on input } s &\in \{0, 1\}^r \text{ at } E_S \\
    &\text{output } (s_1, \ldots, s_r) := s \\
    \text{on input } (j, \text{getkey}) &\text{ at } E_N \\
    &\text{if } s_j = 0 \text{ then output } (j, \bullet) \text{ at } E_N \\
    &\text{else output } (j, k_j) \text{ at } E_N
\end{align*}
\]

Example 3.2 (Secure Channel). The core resource $\text{SEC}^r$ for $r$ sessions also takes as switch value at interface $E_S$ an $r$-bit string which specifies for each session whether or not confidentiality is “broken”. The resource $\text{SEC}^r$ allows Alice to send one message per session to Bob. Eve learns nothing about the transmitted message but its length, unless this session was marked as “broken”, in which case the message is leaked to her. The channel $\text{SEC}^r$ does not allow Eve to inject any message, regardless of the value of the switch, and is formalized in System 3.2.

Example 3.3 (Local and Global Password-Guessing Triggers). Eve will not be allowed to influence the switch values of $\text{KEY}^r$ or $\text{SEC}^r$ directly, instead she will have to interact with a trigger system which captures the guessing of per-session passwords. We consider two different such trigger systems, in both of them the number of guesses allowed to Eve is restricted. (This restriction can be formalized by a monotone binary output which becomes 1 as soon as the number of guesses is exhausted to formally obtain a restricted system as described in Section 2.4.3.) These two systems

---

$^1$ To match the formal definition of a random system described in Section 2.4, which provides an output for each received input, the core resources $\text{KEY}^r$ and $\text{SEC}^r$ should output a dummy message at the $E_N$-interface every time a switch value is input at the $E_S$-interface. This technicality is omitted here and below for the sake of clarity.
3.1 Transformable Systems

**System 3.2: Core resource SEC**

\[ s_j := 0 \text{ and } m_j := \diamond, \text{ for all } j \in \{1, \ldots, r\} \]

**on first input** \((j, m)\) at A

\[
\begin{align*}
m_j &:= m \\
\text{output } (j, m_j) &\text{ at B} \\
\text{output } (j, |m_j|) &\text{ at } E_N
\end{align*}
\]

**on input** \(s \in \{0,1\}^r\) at \(E_S\)

\[
(s_1, \ldots, s_r) := s
\]

**on input** \((j, \text{getmsg})\) at \(E_N\)

\[
\begin{align*}
\text{if } s_j = 0 \text{ then output } (j, \bullet) &\text{ at } E_N \\
\text{else output } (j, m_j) &\text{ at } E_N
\end{align*}
\]

differ in whether the restriction on the number of guesses is local to each session or global over all \(r\) sessions. We refer to them as local and global (password-guessing) triggers and denote them by LT and GT, respectively.

Formally, both triggers are parameterized by a password distribution \(\mathcal{P}\) over \(W^r\) (where \(W \subseteq \{0,1\}^*\) is a set of passwords) and the number of password guesses allowed, either locally for each of the sessions (a tuple \(q := (q_1, \ldots, q_r)\)) or globally (a parameter \(q\)). Both LT \((\mathcal{P}, q)\) and GT \((\mathcal{P}, q)\) initially sample \(r\) passwords \((w_1, \ldots, w_r)\) according to \(\mathcal{P}\). When a password guess \((j, w)\) for the \(j^{th}\) session is received, LT \((\mathcal{P}, q)\) changes the state of this session to “broken” if the password guess is correct and no more than \(q_j\) guessing queries were made to that session. In contrast, GT \((\mathcal{P}, q)\) declares a session “broken” if the password was correctly guessed for this session and no more than \(q\) password-guessing queries were made in total to all sessions. Both triggers LT \((\mathcal{P}, q)\) and GT \((\mathcal{P}, q)\) are only accessible by Eve and are detailed in Systems 3.3 and 3.4.

Combining the core systems and triggers given above via Definition 3.1 leads to four downgradable resources:

- two with local restrictions, \(\text{KEY}^r_{\text{LT}(\mathcal{P}, q)}\) and \(\text{SEC}^r_{\text{LT}(\mathcal{P}, q)}\), where the number of password-guessing queries is restricted per session;
- two with a global restriction, \(\text{KEY}^r_{\text{GT}(\mathcal{P}, q)}\) and \(\text{SEC}^r_{\text{GT}(\mathcal{P}, q)}\), where only the total number of password-guessing queries is limited.

To simplify the notation, we will often drop the parameters \(\mathcal{P}, q, q\) when clear from the context.
3.2 Password-Based Key Derivation

In this section we formalize the simple protocol for deriving a key from a password via hashing that was considered as an example in Section 3.1. We confirm that, as expected, depending on the assumed resources to start with, this protocol constructs one of the variants of the downgradable key resource. These constructions will turn out to be useful to us later.

Informally, the assumed resources our construction starts with consist of a shared password and a hash function (modelled as a random oracle) for each of the sessions. Note that these independent random oracles can be constructed from a single one via salting (i.e., domain separation), a point that we will discuss in greater detail later in Section 3.4.4.

More formally, we model the shared passwords as an explicit resource denoted PW. It is parameterized by a joint distribution $\mathcal{P}$ of $r$ passwords. The resource $\text{PW}(\mathcal{P})$ first samples from the distribution $\mathcal{P}$ to obtain $r$ passwords $(w_1, \ldots, w_r)$ and then outputs $(j, w_j)$ at interface $i \in \{A, B\}$ whenever it receives as input $(j, \text{getpwd})$ at the same interface $i$. Note that Eve does not learn anything about the sampled passwords excepted for the a priori known distribution $\mathcal{P}$.

Each hash function is modelled as a random oracle available to all parties, denoted by RO. Notably, we model the restriction on Eve’s computational power by a restriction on the number of invocations of the random oracles that she is allowed to do. (For a rationale behind this choice and how it allows to model complexity amplification via iteration, see Chapter 5.) We consider either a tuple of random oracles with local restrictions denoted $[\text{RO}_{q_1}, \ldots, \text{RO}_{q_r}]$, where each random oracle has its own upper bound $q_j$ on the number of adversarial queries it allows; or a tuple of random oracles with one global restriction denoted $[\text{RO}, \ldots, \text{RO}]_{q}$, where at most $q$ adversarial queries are allowed in total.
Our key-derivation protocol is performed by a converter $kd$. Upon a
key request $(j, \text{getkey})$ for the $j^{\text{th}}$ session, it queries $PW(\mathcal{P})$ to retrieve the
shared password $w_j$ for this session, then queries the $j^{\text{th}}$ random oracle on
$w_j$ and returns its output.

The following simple lemma shows that the protocol $kd := (kd, kd)$,
where each party simply applies the converter $kd$, allows users to obtain
downgradable keys in the sense of Section 3.1.2.

**Lemma 3.1.** For the key derivation protocol $kd := (kd, kd)$ described
above, there exists a simulator $\sigma_{kd}$ such that for all distributions $\mathcal{P}$ of $r$
passwords, for all integers $q := (q_1, \ldots, q_r)$ and $q$, we have

\[
\begin{bmatrix} \text{RO}_{q_1}, \ldots, \text{RO}_{q_r} \end{bmatrix}, PW(\mathcal{P}) \xrightarrow{\text{ABE}} \text{KEY}_{\text{LT}(\mathcal{P}, q)}^{r}, \text{KEY}_{\text{GT}(\mathcal{P}, q)}^{r}
\]

**Proof.** The simulator $\sigma_{kd}$ emulates $r$ random oracles mostly by lazy sam-
pling. More precisely, upon an adversarial query $x$ made to the $j^{\text{th}}$ random
oracle, the simulator $\sigma_{kd}$ forwards $x$ as a password guess for the $j^{\text{th}}$ session
to the trigger $\text{LT}(\mathcal{P}, q)$ or $\text{GT}(\mathcal{P}, q)$ at its $\text{in}_S$ interface and then tries to
retrieve the key associated with that session by querying its $\text{in}_N$-interface.
If the password guess was correct, then the simulator $\sigma_{kd}$ can output the
retrieved key as a simulated output of the random oracle, and otherwise
it just samples a uniform $n$-bit string. The simulator $\sigma_{kd}$ is described in
System 3.5.

Note that the simulator $\sigma_{kd}$ does not have to keep track of the number
of queries since this is handled directly by the triggering system which
receives every query received by $\sigma_{kd}$. The simulation is therefore perfect in
both settings, independently of whether the random oracles are locally or
globally restricted. \hfill \Box

### 3.3 Password-Based Message Authentication

In this section we investigate the use of password-derived keys for message
authentication using MACs. We prove that such a construction meets the
intuitive expectation that in a multi-user setting, as long as a password for
a particular session is not guessed, the security (in this case: authenticity)
in that session is maintained at the same level as if a perfectly random key
was used. We refer to Figure 3.2 for the depictions of the real and the ideal
experiment involved in the construction statement.
System 3.5: Simulator $\sigma_{kd}$

\[ g_j(x) := \Diamond, \text{ for all } x \in \{0, 1\}^* \text{ and } j \in \{1, \ldots, r\} \]

on input $(j, x)$ at $\text{out}_1$

output $(j, x)$ at $\text{in}_S$

if $g_j(x) = \Diamond$ then

$(j, k) := $ result of querying $(j, \text{getkey})$ at $\text{in}_N$

if $k \neq \Diamond$ then $g_j(x) := k$

else $g_j(x) \leftarrow \{0, 1\}^n$

output $(j, g_j(x))$ at $\text{out}_1$

---

**Figure 3.2.** Left: The assumed resource, a downgradable key $\text{KEY}^r_T$ and an insecure channel $\text{INSEC}^r_r$, with protocol converters $\text{tag}$ and $\text{vrf}$ attached to interfaces $A$ and $B$, denoted $\text{tag}^A\text{vrf}^B[\text{KEY}^r_T, \text{INSEC}^r_r]$. Right: The desired downgradable unordered authenticated channel $\text{UAUT}^r_T$ with simulator $\sigma_{\text{mac}}$ attached to interface $E$, denoted $\sigma_{\text{mac}}^E\text{UAUT}^r_T$. The simulator $\sigma_{\text{mac}}$ must emulate Eve’s interface in the left picture, i.e., key retrieval queries at $E_{1,N}$, trigger queries at $E_{1,S}$ and the insecure channel at $E_2$. 
3.3 Password-Based Message Authentication

We present these results partly to put them in contrast with our later findings in Section 3.4 on password-based encryption, where the situation turns out to be more intricate.

**Assumed resources.** In the construction statement shown below, we assume the availability of a password-derived key and an insecure communication channel for each of the \( r \) considered sessions. For password-derived keys, we simply use the downgradable resource \( \text{KEY}_T \) which can be constructed e.g. via one of the statements in Lemma 3.1 (here \( T \) stands for either \( \text{LT} (P, q) \) or \( \text{GT} (P, q) \)). The insecure channels are formalized as the resource \( \text{INSEC}^r \) which forwards any message sent by Alice directly to Eve, while any message injected by Eve is forwarded directly to Bob.

**MAC schemes as protocols.** Recall the definition of a MAC scheme stated in Section 2.6.2. Given a MAC scheme \( (\text{tag}, \text{vrf}) \), the protocol for Alice and Bob works in the natural way (we denote their converters \( \text{tag} \) and \( \text{vrf} \), respectively). When \( \text{tag} \) receives as input a message \( m \) for the \( j \)th session consisting of a pair \((j, m)\), it retrieves the key \( k_j \) associated to this session from the downgradable key resource \( \text{KEY}_T \), computes the \( \text{tag} u := \text{tag}(k_j, m) \) and outputs to the insecure channel \( \text{INSEC}^r \) the pair \((j, m \parallel u)\). On the other end of the channel, whenever \( \text{vrf} \) receives a message and a tag for some session, consisting of a pair \((j', m' \parallel u')\), it first retrieves the key \( k_{j'} \) associated to this session from \( \text{KEY}_T \), computes \( \text{vrf}(k_{j'}, m', u') \) and outputs \((j', m')\) only if the verification succeeds.

**Constructed resource.** The channel that Alice and Bob obtain by using the protocol \( (\text{tag}, \text{vrf}) \) guarantees that any message that Bob receives for a particular session must have been sent before by Alice, unless this session was “broken”. This *core unordered authenticated channel*, denoted \( \text{UAUT}^r \), thus takes an \( r \)-bit string \((s_1, \ldots, s_r)\) as a switch value, specifying for each session \( j \) whether it is broken \((s_j = 1)\), in which case Eve can send any message to Bob for this particular session, or not \((s_j = 0)\), in which case the messages that Eve can send to Bob for session \( j \) are limited to those that Alice already sent. The channel \( \text{UAUT}^r \) does not offer any secrecy, every message input by Alice is directly forwarded to Eve independently of the current switch value, while the switch value \( s_j \) of a particular session can also be retrieved by Eve. Note that the unordered authenticated channel \( \text{UAUT}^r \), similarly to a MAC scheme, only aims to prevent Eve from being able to inject a *fresh* message, it does not a priori prevent the injection of a legitimate message multiple times, the reordering
of legitimate messages, or the loss of some messages. The channel $\text{UAUT}^r$ is described more precisely in System 3.6.

**System 3.6: Channel $\text{UAUT}^r$**

\[
\begin{align*}
& s_j := 0 \text{ and } Q_j := \emptyset, \text{ for all } j \in \{1, \ldots, r\} \\
& \text{on input } (j, m) \text{ at } A \\
& \quad Q_j := Q_j \cup \{m\} \\
& \quad \text{output } (j, m) \text{ at } E_N \\
& \text{on input } s \in \{0, 1\}^r \text{ at } E_S \\
& \quad (s_1, \ldots, s_r) := s \\
& \text{on input } (j', m') \text{ at } E_N \\
& \quad \text{if } (s_{j'} = 1) \lor (m' \in Q_{j'}) \text{ then } \\
& \quad \text{output } (j', m') \text{ at } B
\end{align*}
\]

The next theorem states that if the MAC scheme used by the protocol $(\text{tag}, \text{vrf})$ is weakly unforgeable, then it constructs the downgradable unordered authenticated channel $\text{UAUT}^r_T$ by using the downgradable key $\text{KEY}^r_T$ and the insecure channel $\text{INSEC}^r$. The arguments used in the proof of Theorem 3.1 could easily be extended to the case of any trigger $T$ with output space $\{0, 1\}^r$ and bit-wise non decreasing output.

**Theorem 3.1.** Let $\text{MAC} := (\text{tag}, \text{vrf})$ be a correct MAC scheme and consider the associated protocol $\text{mac} := (\text{tag}, \text{vrf})$. Then, there exists a simulator $\sigma_{\text{mac}}$ described in System 3.7 such that for every distribution $P$ of $r$ passwords, every tuple of $r$ integers $q := (q_1, \ldots, q_r)$ and any integer $q$, and any trigger $T \in \{\text{LT} (P, q), \text{GT} (P, q)\}$,

\[
[\text{KEY}^r_T, \text{INSEC}^r] \xrightarrow{\text{(mac, } \sigma_{\text{mac}}, \epsilon_T)} \text{UAUT}^r_T,
\]

where $\epsilon_T (D) := r \cdot \Gamma^{\text{CMA}}_T (G^{\text{CMA}} (\text{MAC}))$, for every distinguisher $D$, where the reduction system $\text{C}^{\text{CMA}}_T$ depends on the trigger $T$ and is described below in the proof.

**Proof.** According to Definition 2.5, we need to find a simulator $\sigma_{\text{mac}}$ and show that the systems $\text{tag}^A \text{vrf}^B [\text{KEY}^r_T, \text{INSEC}^r]$ and $\sigma_{\text{mac}}^E \text{UAUT}^r_T$ are indistinguishable. The role of the simulator $\sigma_{\text{mac}}$ is to emulate what happens at Eve’s interface $E$ in the real system $\text{tag}^A \text{vrf}^B [\text{KEY}^r_T, \text{INSEC}^r]$ by having only access to the idealized channel $\text{UAUT}^r_T$. The simulator $\sigma_{\text{mac}}$ has therefore two tasks: 1) to emulate key retrieval queries and password-guessing queries intended for the key resource $\text{KEY}^r_T$; and 2) to emulate messages injected by Eve to the insecure channel $\text{INSEC}^r$, as well as messages output by $\text{INSEC}^r$ to Eve.
3.3 Password-Based Message Authentication

In the real system, a message \((j, m)\) input by Alice for session \(j\) is seen by Eve as \((j, m \parallel u)\), where \(u := \text{tag}(k_j, m)\) and \(k_j\) is a key which was initially selected uniformly at random. In the ideal system, such a message \((j, m)\) is directly output by the channel \(\text{UAUT}_T^r\) to the simulator which therefore needs to mimic the tagging process. The simulator \(\sigma_{\text{mac}}\) can easily do so by initially selecting a key \(k_j\) uniformly at random and outputting \((j, m \parallel \text{tag}(k_j, m))\). The simulator \(\sigma_{\text{mac}}\) forwards any password-guessing query to the trigger \(T\) of the channel \(\text{UAUT}_T^r\), whereas \(\sigma_{\text{mac}}\) deals with key retrieval queries for session \(j\) by first asking the channel \(\text{UAUT}_T^r\) whether session \(j\) is broken, and then outputting accordingly the key \(k_j\) used to simulate the tagging process or the error symbol \(\bullet\).

Note that so far the simulation is perfect. The only difference between the real and ideal system lies in the way injection queries are handled. An injection message \((j', m' \parallel u')\) made by Eve into the real system is output to Bob as \((j', m')\) only if the tag is valid, i.e., \(\text{vrf}(k_{j'}, m', u') = 1\). Thus, when the simulator receives such a query it only outputs \((j', m')\) to the channel \(\text{UAUT}_T^r\) if the tag \(u'\) is valid. As a result, the channel \(\text{UAUT}_T^r\) outputs the desired message \((j', m')\) to Bob, unless the session \(j'\) is not broken and this message was never input by Alice. The simulator \(\sigma_{\text{mac}}\) is described in more details in System 3.7.

### System 3.7: Simulator \(\sigma_{\text{mac}}\)

\[
\begin{align*}
\text{for all } j \in \{1, \ldots, r\} & \quad k_j \leftarrow \{0, 1\}^n \quad \text{on input } (j, m) \text{ at } \text{in}_N \quad \text{output } (j, m \parallel u) \text{ at } \text{out}_2 \\
\text{on input } (j', m' \parallel u') \text{ at } \text{out}_2 & \quad \text{if } \text{vrf}(k_{j'}, m', u') = 1 \text{ then } \text{output } (j', m') \text{ at } \text{in}_N \\
\text{on input } (j, w) \text{ at } \text{out}_{1,S} & \quad \text{output } (j, w) \text{ at } \text{in}_S \\
\text{on input } (j, \text{getkey}) \text{ at } \text{out}_{1,N} & \quad s := \text{result of querying } (j, \text{getsw}) \text{ at } \text{in}_N \\
& \quad \text{if } s = 1 \text{ then output } (j, k_j) \text{ at } \text{out}_{1,N} \\
& \quad \text{else output } (j, \bullet) \text{ at } \text{out}_{1,N}
\end{align*}
\]

In order to have shorter notations within the proof, the real system \(\text{tag}^A \text{vrf}^B [\text{KEY}_T^r, \text{INSEC}_r]\) will be denoted by \(R_T\) and the ideal system \(\sigma_{\text{mac}}^E \text{UAUT}_T^r\) will be denoted by \(S_T\), where \(T \in \{\text{LT}(\mathcal{P}, q), \text{GT}(\mathcal{P}, q)\}\).
Observe that the only difference between the systems $R_T$ and $S_T$ is that if a fresh message $m'$ is injected by Eve together with a successfully forged tag, such a message $m'$ is always output at Bob’s interface $B$ in the system $R_T$, whereas in $S_T$ such a message $m'$ is only output if the password associated with that session was previously guessed. Let us add a monotone binary output to the systems $R_T$ and $S_T$, to obtain the corresponding games $\overline{R}_T$ and $\overline{S}_T$, defined as follows: the MBO $A_1, A_2, \ldots$ becomes 1 as soon as $(j, m' \parallel u')$ is input to the $E_2$-interface such that the message $(j, m')$ was never input to the $A$-interface before, the tag is valid $vrf (k_j, m', u') = 1$, and the $j^{th}$ password was not guessed ($s_j = 0$). Then, the previous observation implies that $\overline{R}_T \equiv \overline{S}_T$ and together with Lemma 2.5 we have

$$\Delta^D (R_T, S_T) \leq \Gamma^D (\overline{R}_T),$$

for all distinguishers $D$.

System 3.8: Reduction $C_{T,j}^{\text{CMA}}$

\[
(s_1, \ldots, s_r) := 0^r \text{ and } k_1, \ldots, k_r \leftarrow \{0,1\}^n \\
\text{on input } (j, m) \text{ at } A \\
\quad u \leftarrow \text{tag} (k_j, m) \\
\quad \text{if } j = j^* \text{ then } u := \text{result of querying } (\text{tag}, m) \text{ at in} \\
\quad \text{output } (j, m \parallel u) \text{ at } E_2 \\
\text{on input } (j', m' \parallel u') \text{ at } E_2 \\
\quad b := \text{vrf} (k_j, m', u') \\
\quad \text{if } j' = j^* \text{ then } b := \text{result of querying } (\text{vrf}, m', u') \text{ at in} \\
\quad \text{if } b = 1 \text{ then output } (j', m') \text{ at } B \\
\text{on input } (j, w) \text{ at } E_{1,S} \\
\quad (s_1, \ldots, s_r) := \text{emulate result of querying } T \text{ on } (j, w) \\
\text{on input } (j, \text{getkey}) \text{ at } E_{1,N} \\
\quad \text{if } (s_j = 0) \lor (j = j^*) \text{ then output } (j, \bullet) \text{ at } E_{1,N} \\
\quad \text{else output } (j, k_j) \text{ at } E_{1,N}
\]

We now reduce the task of winning the game $\overline{R}_T$ to the task of winning the WUF-CMA security game $G^{\text{CMA}} (\text{MAC})$ described in System 2.1 in Section 2.6.2. To do so, we consider a sequence of reduction systems $C_{T,1}^{\text{CMA}}, \ldots, C_{T,r}^{\text{CMA}}$, where each reduction $C_{T,j}^{\text{CMA}}$ has 5 outside sub-interfaces (labeled $A, B, E_{1,N}, E_{1,S}$ and $E_2$) and connects at its inside interface to the WUF-CMA game $G^{\text{CMA}} (\text{MAC})$, for all $j^* \in \{1, \ldots, r\}$. The basic idea of
the reduction $\mathbf{C}^\text{CMA}_{T,j^*}$ is to forward any tag or verification query for session $j^*$ to the system connected at its inside interface, while other sessions are handled locally. The reduction $\mathbf{C}^\text{CMA}_{T,j^*}$ also emulates internally the trigger $T$, which in the case of $\text{LT}(P,q)$ or $\text{GT}(P,q)$ corresponds to first sample $r$ passwords according to $P$ and then to appropriately count the number of password-guessing queries, in order to keep track of which session should be considered “broken”. Key retrieval queries are handled as usual, excepted for session $j^*$ for which the error symbol $*$ is always returned. A more formal description of the reduction system $\mathbf{C}^\text{CMA}_{T,j^*}$ is given in System 3.8.

Consider a distinguisher $\mathcal{D}$ interacting with the game $\overline{R}_T$ in order to provoke its MBO. Let $\mathcal{F}_j$ be the event that in this random experiment the distinguisher $\mathcal{D}$ won the game $\overline{R}_T$ by finding a forgery for a fixed session $j$, i.e., $(j,m' \parallel u')$ was input at the $E_2$-interface such that $(j,m')$ was never input to the $A$-interface before, the tag is valid $\text{vrf}(k_j,m',u') = 1$, and the $j$th password was not guessed ($s_j = 0$), for some message $m' \in \mathcal{M}$ and tag $u' \in \mathcal{U}$. The probability that this event $\mathcal{F}_j$ happens in this random experiment is denoted $\Pr_{\mathcal{D}}^{\overline{R}_T}(\mathcal{F}_j = 1)$. Note that winning the game $\overline{R}_T$ involves provoking one of the events $\mathcal{F}_j$, for some session $j \in \{1, \ldots, r\}$, and thus

$$\Gamma^\mathcal{D}(\overline{R}_T) \leq \sum_{j \in \{1, \ldots, r\}} \Pr_{\mathcal{D}}^{\overline{R}_T}(\mathcal{F}_j = 1).$$

Let us fix the randomness used in the random experiment $\mathcal{D} \overline{R}_T$ (consisting of the randomness of the distinguisher $\mathcal{D}$, the $r$ keys used for tagging, the $r$ passwords and the randomness of the algorithm $\text{tag}$). Conditioned on this randomness, any transcript of queries which provokes the event $\mathcal{F}_j$ in $\mathcal{D} \overline{R}_T$ also wins the game $\mathbf{C}^\text{CMA}_{T,j\overline{G}^\text{CMA}}(\text{MAC})$. Moreover, in a such a transcript the password associated with the $j$th session is not guessed ($s_j = 0$) and thus such a transcript happens with the same probability in the random experiment $\mathbf{D} \mathbf{C}^\text{CMA}_{T,j\overline{G}^\text{CMA}}(\text{MAC})$. Hence,

$$\Pr_{\mathcal{D}}^{\overline{R}_T}(\mathcal{F}_j = 1) \leq \Gamma^\mathbf{C}^\text{CMA}_{T,j}(\mathbf{G}^\text{CMA}(\text{MAC})), $$

for all sessions $j \in \{1, \ldots, r\}$. The previous equations implies that the reduction $\mathbf{C}^\text{CMA}_T$, which consists in selecting a session $j^*$ uniformly at random in $\{1, \ldots, r\}$ and then implementing the reduction $\mathbf{C}^\text{CMA}_{T,j^*}$ is such that

$$\Gamma^\mathcal{D}(\overline{R}_T) \leq r \cdot \Gamma^\mathbf{C}^\text{CMA}_{T,j}(\mathbf{G}^\text{CMA}(\text{MAC})), $$

for all distinguishers $\mathcal{D}$. 

$\Box$
3.4 Password-Based Encryption

In this section we investigate the use of password-derived keys for symmetric encryption. In a multi-session setting, one would again intuitively expect that as long as a password for a particular session is not guessed, the confidentiality in that session is maintained. This would, roughly speaking, correspond to a construction of (downgradable) secure channels from authenticated channels and password-derived keys. We now describe this desired construction in greater detail, referring to Figure 3.3 for the depictions of the real and the ideal experiment.

**Figure 3.3.** Left: The assumed resource, a downgradable key $\text{KEY}_T^r$ and an authenticated channel $\text{AUT}_T^r$, with protocol converters $\text{enc}$ and $\text{dec}$ attached to interfaces A and B, denoted $\text{enc}^A \text{dec}^B [\text{KEY}_T^r, \text{AUT}_T^r]$. Right: The desired downgradable secure channel $\text{SEC}_T^r$ with simulator $\sigma$ attached to interface E, denoted $\sigma^E \text{SEC}_T^r$. The simulator $\sigma$ must emulate Eve’s interface in the left picture, i.e., key retrieval queries at $E_{1,N}$, trigger queries at $E_{1,S}$ and the authenticated channel at $E_2$.

**Assumed resources.** These construction statements assume the availability of a password-derived key and an authenticated communication channel for each of the $r$ sessions. For password-derived keys, we use the downgradable resource $\text{KEY}_T^r$, where $T$ typically stands for either $\text{LT}(P, q)$ or $\text{GT}(P, q)$. The authenticated channel is formalized as a resource $\text{AUT}_T^r$ which takes as input a pair $(j, c)$ at Alice’s interface A, corresponding to a ciphertext $c$ to be transmitted for the $j^{th}$ session, and outputs $(j, c)$ at both Eve’s interface E and Bob’s interface B. For convenience, we will also assume that at most one message is transmitted per session, but our results could easily be extended to the case of multiple messages per session. We also simplify our statements by assuming that the channel $\text{AUT}_T^r$ has no
3.4 Password-Based Encryption

particular weaknesses, i.e., Eve can neither replay messages nor drop those coming from Alice.

**Encryption schemes as protocols.** Recall the definition of an encryption scheme stated in Section 2.6.3. Given an encryption scheme \((\text{enc}, \text{dec})\), the protocol for Alice and Bob again works in the natural way (their converters are denoted \(\text{enc}\) and \(\text{dec}\), respectively). Both \(\text{enc}\) and \(\text{dec}\) expect two resources at their respective inside interface \(\text{in}\): a downgradable key resource \(\text{KEY}_T\) at their interface \(\text{in}_1\) and an authenticated communication channel \(\text{AUT}_T\) at their interface \(\text{in}_2\). The converter \(\text{enc}\) takes as input a pair \((j, m)\) at its outside interface \(\text{out}\), corresponding to a message \(m\) to be transmitted for the \(j\)th session. It retrieves the shared secret key \(k_j\) for this session by inputting \((j, \text{getkey})\) at its \(\text{in}_1\)-interface, computes the ciphertext \(c \leftarrow \text{enc}(k_j, m)\) and finally outputs \((j, c)\) to the channel at its \(\text{in}_2\)-interface. Similarly, when a pair \((j, c)\) is input to the \(\text{in}_2\)-interface of the converter \(\text{dec}\), corresponding to an encrypted message sent for the \(j\)th session, the converter \(\text{dec}\) retrieves the shared secret key \(k_j\) for this session (by inputting \((j, \text{getkey})\) at its \(\text{in}_1\)-interface) and outputs \((j, \text{dec}(k_j, c))\) at its \(\text{out}\)-interface. Throughout this section, we will assume the underlying encryption scheme \((\text{enc}, \text{dec})\) to be correct.

**Desired resource.** The channel that Alice and Bob wish to obtain by using the protocol \(\text{se} := (\text{enc}, \text{dec})\) is the downgradable resource \(\text{SEC}_T\) described in Section 3.1.2, which guarantees that any message sent by Alice for a particular session is transmitted confidentially to Bob, unless this session was “broken”.

The goal of password-based encryption can then be loosely phrased as achieving the following construction

\[
[\text{KEY}_T, \text{AUT}_T] \xrightarrow{(\text{se}, \sigma, \varepsilon)} \text{SEC}_T,
\]

for some simulator \(\sigma\) and “reasonable” distinguishing advantage \(\varepsilon\).

### 3.4.1 PBE for a Single Session

We start by focusing on PBE with a single session. This will serve as an introduction to the study of the multi-session setting, given in Sections 3.4.2 and 3.4.3.

In the particular case of a single session, the local password-guessing trigger \(\text{LT}(P, q)\) and the global one \(\text{GT}(P, q)\) are actually identical for any
distribution $\mathcal{P}$ of a single password and any number $q$ of password-guessing queries. To be consistent with what lies ahead in Section 3.4.3, we will hence always mention $\text{LT} (\mathcal{P}, q)$ in this section. The considered special case also allows us to drop the exponent $r$ indicating the number of sessions and the index $j$ of the session associated with any input or output, thus simplifying the notation.

We are interested in the possibility of constructing the downgradable secure channel $\text{SEC}_{\text{LT}} (\mathcal{P}, q)$ from a downgradable key $\text{KEY}_{\text{LT}} (\mathcal{P}, q)$ and an authenticated channel $\text{AUT}$ using the protocol $\text{se} = (\text{enc}, \text{dec})$. According to Definition 2.5 we must thus find a simulator $\sigma$ such that the systems $\text{enc}^A \text{dec}^B \left[ \text{KEY}_{\text{LT}} (\mathcal{P}, q), \text{AUT} \right]$ and $\sigma^E \text{SEC}_{\text{LT}} (\mathcal{P}, q)$ represented in Figure 3.3 with $T = \text{LT} (\mathcal{P}, q)$, are indistinguishable.

The commitment problem. In the real world, whenever a message $m$ is input at Alice’s interface $A$, the corresponding ciphertext is output at Eve’s interface $E_2$. On the other hand, in the ideal world only the length $|m|$ of the transmitted message $m$ is output by the channel $\text{SEC}_{\text{LT}} (\mathcal{P}, q)$ to the simulator $\sigma$. The simulator must therefore emulate that a ciphertext was sent by only knowing the length $|m|$ of the transmitted message and not the message $m$ itself.

A naïve simulation strategy could start as follows. The simulator $\sigma$ initially selects a key $k$ uniformly at random and emulates the transmission of a ciphertext by encrypting a fresh random message $v$ of the correct length under key $k$, while password-guessing queries are simply forwarded to the trigger $\text{LT} (\mathcal{P}, q)$ of the downgradable channel $\text{SEC}_{\text{LT}} (\mathcal{P}, q)$.

However, this approach does not work. To see this, consider what happens when the password is guessed and the session is broken (which, depending on $\mathcal{P}$, may happen with a large probability). In the real world, the distinguisher can retrieve the key $k$ used for encryption and check that the previously seen ciphertext $c$ is indeed an encryption of the transmitted message $m$. In contrast, in the ideal world the simulator $\sigma$ can retrieve the transmitted message $m$, but note that it cannot output the key $k$ that it chose at the beginning to simulate encryption since $\text{dec} (k, c) = v$ is a random message which (with overwhelming probability) is different from the actual transmitted message $m$. The simulator $\sigma$ must therefore “decommit” by finding a key $k'$ such that the decryption of the simulated ciphertext $c$ under that key $k'$ yields the transmitted plaintext $m$, i.e., $\text{dec} (k', c) = m$. However, it is not hard to see that unless the key space of the encryption scheme contains as many keys as there are messages (which is only true for impractical schemes such as the one-time pad), it is highly
likely that such a key even exists and the simulation therefore fails.

**Brute-force to the rescue.** The previous paragraph only shows that one particular simulation strategy fails. The source of the commitment problem is that the simulator $\sigma$ only breaks the session after having output the simulated ciphertext. The key insight is that this does not have to be the case.

To prevent the simulation breakdown, we instead consider a different simulator $\sigma_{LT}$ which, in contrast to $\sigma$, tries to break the session before having to output any ciphertext. So, instead of faithfully forwarding the $q$ password-guessing queries, the simulator $\sigma_{LT}$ initially exhausts all of the allowed $q$ queries to optimally brute-force the session by querying the $q$ most likely passwords. If this initial brute-force step fails, $\sigma_{LT}$ simply encrypts a random message of the correct length and declares any password guess as incorrect so that the actual key used to simulate the encryption is never output. However, if the initial brute-force step succeeds, $\sigma_{LT}$ has access to the transmitted message and can therefore perfectly simulate the corresponding ciphertext, while password-guessing queries can be appropriately “scaled up” to match the guessing probability of the real world. In this case, the key used to simulate encryption can obviously be output since the ciphertext produced is an actual encryption of the transmitted message.

In this setting with a single session, password-based encryption is therefore possible with respect to the simulation strategy $\sigma_{LT}$ sketched above.

**Corollary (Informal).** For every distribution $\mathcal{P}$ of a single password and every integer $q$, there exists a simulator $\sigma_{LT}$ such that

$$\left[\text{KEY}_{LT}(\mathcal{P},q), \text{AUT}\right] \xrightarrow{\text{SE}_{\text{LT}(\mathcal{P},q)}} \text{SEC}_{\text{LT}(\mathcal{P},q)},$$

where the distinguishing advantage $\varepsilon$ can be reduced to the IND-CPA security of the underlying encryption scheme.

The generalization of the above statement for multiple sessions and its proof, together with the definition of the simulator $\sigma_{LT}$, are postponed to Theorem 3.3 in Section 3.4.3. This corollary then easily follows from Theorem 3.3 by taking $r = 1$.

### 3.4.2 General Impossibility of PBE

In this section we show that the above positive result concerning password-based encryption for a single session can in general not be lifted to the
case of multiple sessions. Our impossibility result consists of providing a lower bound on the distinguishing advantage of a particular distinguisher $D_\ell$ in distinguishing the systems $\text{enc}^A \text{dec}^B [\text{KEY}_T, \text{AUT}^r]$ and $\sigma^E \text{SEC}^r_T$ depicted in Figure 3.3, for any trigger system $T$ with output space $\{0, 1\}^r$ and any simulator $\sigma$. The lower bound obtained depends on the properties of the trigger system $T$ and while giving a clear impossibility result for some triggers, for others it becomes moot. In particular, while it gives a strong bound for the case of the global password-guessing trigger $\text{GT}(P, q)$, the bound is inconclusive for the local trigger $\text{LT}(P, q)$ and independently distributed passwords. We show later in Section 3.4.3 that in this specific case password-based encryption is actually possible.

The core of our impossibility result lies in exploiting the commitment problem explained in Section 3.4.1. Intuitively, the simulator $\sigma$ can avoid this commitment problem by trying to break the session associated with the plaintext before having to output the corresponding ciphertext. This works out if $\sigma$ follows the optimal strategy for breaking this particular session, since an arbitrary distinguisher would no be able to do better. However, since $\sigma$ does not a priori know which session will have to be “decommitted”, the simulator $\sigma$ must be able to follow the optimal strategy for each session. This might be possible depending on the trigger system $T$ (such as in the case of $\text{LT}(P, q)$ if the passwords are independently distributed), but in general following the optimal strategy for a particular session may prevent $\sigma$ from following the optimal strategy for another session. This is for example the case for the trigger $\text{GT}(P, q)$ where following the optimal strategy for a particular session consists of exhausting all the $q$ allowed password-guessing queries on this session.

The high level idea of the distinguisher $D_\ell$ is therefore to first force the simulator to be committed to a ciphertext in every session; and second, to pick a session $j^*$ uniformly at random and to follow the optimal strategy to break it. To avoid the commitment problem, the simulator must in contrast try to break the maximum number of sessions before simulating the ciphertexts since it does not know which session $j^*$ will be chosen by the distinguisher.

We need to add some notation in order to quantify more precisely the distinguishing advantage achieved by $D_\ell$. Let $T$ denote an arbitrary trigger system with output space $\{0, 1\}^r$, and consider a distinguisher $D$ interacting with the trigger $T$ alone. Note in particular that such a distinguisher sees the output $(s_1, \ldots, s_r) \in \{0, 1\}^r$ of $T$. The interaction of the distinguisher $D$ with the trigger $T$ defines a random experiment, and we denote by $G_j$ the event that the $j^{th}$ session is “broken”, i.e., $T$ output at
least once an \( r \)-bit string \((s_1, \ldots, s_r)\) with \( s_j = 1 \), for all \( j \in \{1, \ldots, r\} \). The probability that this event \( G_j \) happens in this random experiment is denoted \( P^T \left( G_j = 1 \right) \) and it will be convenient to see it as a function of both \( j \) and \( D \). To do so, let \( D \) denote the set of distinguishers for \( T \) and consider the function \( \Gamma^T : \{1, \ldots, r\} \times D \to [0, 1] \), where \( \Gamma^T (j, D) := P^T \left( G_j = 1 \right) \), for all \( j \in \{1, \ldots, r\} \) and \( D \in D \).

In the following, we denote by \( \Gamma^T_{\text{opt}} \) the average of the maximum probability in breaking a particular session, while \( \Gamma^T_{\text{avg}} \) denotes the maximum average probability of breaking all sessions,

\[
\Gamma^T_{\text{opt}} := \frac{1}{r} \sum_{j \in \{1, \ldots, r\}} \max_{D \in D} \Gamma^T (j, D),
\]

\[
\Gamma^T_{\text{avg}} := \frac{1}{r} \max_{D \in D} \sum_{j \in \{1, \ldots, r\}} \Gamma^T (j, D).
\]

(3.1)

Note that \( \max_{D \in D} \Gamma^T (j, D) \geq \Gamma^T (j, D) \), for every \( j \in \{1, \ldots, r\} \), and thus \( \Gamma^T_{\text{opt}} \geq \Gamma^T_{\text{avg}} \). In other words, \( \Gamma^T_{\text{opt}} \) measures the success of an optimal strategy for breaking a \textit{known} randomly chosen session \( j \), while \( \Gamma^T_{\text{avg}} \) measures the success of an optimal strategy for breaking an \textit{unknown} randomly chosen session. Knowing in advance which session will be “attacked” is a clear advantage that we measure by the difference \( \Gamma^T_{\text{opt}} - \Gamma^T_{\text{avg}} \geq 0 \).

**Theorem 3.2.** Let \( SE := (\text{enc}, \text{dec}) \) be a correct encryption scheme with key space \( K := \{0, 1\}^n \) and message space \( M \subseteq \{0, 1\}^* \), and consider the associated protocol \( SE := (\text{enc}, \text{dec}) \). Let \( T \) be a trigger system with output space \( \{0, 1\}^r \) and let \( M_\ell \) denote a non-empty set of messages of fixed length \( \ell \) in \( M \), for some integer \( \ell \). Then, there exists a distinguisher \( D_\ell \) described in System 3.9 such that for all simulators \( \sigma \) we have

\[
\Delta^{D_\ell} \left( \text{enc}^A \text{dec}^B \left[ \text{KEY}^r_T, \text{AUT}^r_T \right], \sigma^E \text{SEC}^{r_E}_T \right) \geq \delta^T - \frac{|K|}{|M_\ell|},
\]

(3.2)

where \( \delta^T := \Gamma^T_{\text{opt}} - \Gamma^T_{\text{avg}} \geq 0 \), with \( \Gamma^T_{\text{opt}} \) and \( \Gamma^T_{\text{avg}} \) defined in (3.1).

**Remark 3.1.** Before proving Theorem 3.2 let us comment on the lower bound obtained in (3.2). If there are almost as many keys as messages, such as in the one-time pad, then the bound in (3.2) becomes trivial. (Note that if the encryption scheme is the one-time pad, then password-based encryption is obviously possible since there is no “commitment” problem: doing the xor of a given ciphertext and of a given message leads to the appropriate key.) However, most encryption schemes used in practice have
significantly less keys than messages and in this case the dominant term in (3.2) is $\delta_T$, which quantifies for a trigger system $T$ the advantage that can be obtained on average by knowing which session will be “attacked”.

For example, $\delta_{GT}(\mathcal{P},q)$ will be large for most distributions $\mathcal{P}$ of $r$ independent passwords, since following the optimal strategy for session $j$, i.e., querying the $q$ most likely passwords (for session $j$), prevents following the optimal guessing strategy for another session as the remaining number of password-guessing queries will be less than $q$. Obviously, if the passwords are cryptographic keys, i.e., they are uniformly distributed over some large space, then the “passwords” are unguessable and $\delta_{GT}(\mathcal{P},q)$ is negligible.

For the local password-guessing trigger $LT(\mathcal{P},q)$ we have $\delta_{LT}(\mathcal{P},q) = 0$ for any number of password-guessing queries $q_1, \ldots, q_r$, since for this trigger each session is truly independent and the optimal strategy can be executed for each session.

Depending on the trigger system $T$, the distinguisher $D_{\ell}$ given in System 3.9 may not necessarily be efficient, since it must be able to implement the optimal strategy for an arbitrary session of its choice. Nonetheless, it is worth mentioning that in the particular case where the trigger system is $GT(\mathcal{P},q)$, computing the optimal guessing strategy for a particular session is “easy” if the passwords are independently distributed. One simply queries the $q$ most likely passwords for this session. Clearly, this assumes that the distinguisher $D_{\ell}$ knows the password distribution $\mathcal{P}$, and is able to sort the passwords according to their respective probability to determine the $q$ most likely for a particular session.

**Proof (of Theorem 3.2).** The main idea of the distinguisher $D_{\ell}$ is to exploit the advantage quantified by $\delta_T$ in knowing in advance which session $j^* \in \{1, \ldots, r\}$ is going to be “attacked”. In order to have shorter notations within the proof, let $R_T$ denote the system $\text{enc}^A \text{dec}^B [\text{KEY}_T, \text{AUT}]$, while $S_T$ denotes $\sigma^E \text{SEC}_T$, for some simulator $\sigma$.

In a first step, the distinguisher $D_{\ell}$ selects $r$ messages of length $\ell$ uniformly at random, one for each session, and sends them at the $A$-interface to observe the corresponding ciphertexts $c_1, \ldots, c_r$ at the $E_2$-interface, which are real encryptions when interacting with $R_T$ or simulated encryptions when interacting with $S_T$ instead. The distinguisher $D_{\ell}$ then selects uniformly at random a particular session $j^* \in \{1, \ldots, r\}$ and tries to break it by “running” the optimal strategy $D_{\text{opt}}(j^*)$ for that particular session, where

$$D_{\text{opt}}(j) := \arg \max_{D \in \mathcal{D}} \Gamma^T(j, D),$$

for all $j \in \{1, \ldots, r\}$. “Running” $D_{\text{opt}}(j^*)$ implies the following three steps.
First, the trigger value \( t \) output by \( D_{\text{opt}}(j^*) \) is forwarded to the interface \( E_{1,S} \), which when interacting with \( R_T \) corresponds to querying the trigger \( T \) on \( t \). Second, the distinguisher \( D_\ell \) outputs 1 if this query \( t \) resulted in breaking the session \( j^* \) and the key retrieved allows to decrypt the ciphertext \( c_{j^*} \) to the original message \( m_{j^*} \). Otherwise, if the session \( j^* \) is still not broken, the distinguisher \( D_\ell \) queries the interface \( E_{1,N} \) to retrieve a switch value \((s_1, \ldots, s_r)\), which when interacting with \( R_T \) corresponds to the last switch value output by \( T \), and feed it to \( D_{\text{opt}}(j^*) \) to obtain the next trigger value. The distinguisher \( D_\ell \) is given in more details in System 3.9.

**System 3.9: Distinguisher \( D_\ell \)**

\[
m_{j} \leftarrow M_\ell \quad \text{and} \quad c_{j} := \diamond, \text{for all} \quad j \{1, \ldots, r\} \\
 j^* := \diamond \\
\text{for } 1 \leq j \leq r \text{ do} \\
| (j, c_{j}) := \text{result at } E_2 \text{ of} \quad (j, m_{j}) \text{ at } A \\
| j^* \leftarrow \{1, \ldots, r\} \\
\text{return } \text{BreakSession}(j^*) \\
\]

**Procedure BreakSession(\( j \))**

\[
t := 1^{st} \text{ output of } D_{\text{opt}}(j) \\
\text{while } t \neq \bullet \text{ do} \\
| \text{output } t \text{ at } E_{1,S} \\
| (s_1, \ldots, s_r) := \text{GetSwitch} \\
| \text{if } s_j = 1 \text{ then} \\
| (j, k') := \text{result of querying} \\
| (j, \text{getkey}) \text{ at } E_{1,N} \\
| \text{if } \text{dec}(k', c_{j}) = m_{j} \text{ then} \\
| \text{return } 1 \\
| \text{else return } 0 \\
| t := \text{emulate result of} \\
| \text{querying } D_{\text{opt}}(j) \text{ on} \\
| (s_1, \ldots, s_r) \\
| \text{return } 0 \\
\]

**Procedure GetSwitch**

\[
(s_1, \ldots, s_r) := 0^r \\
\text{for } 1 \leq j \leq r \text{ do} \\
| (j, k') := \text{result of querying} \\
| (j, \text{getkey}) \text{ at } E_{1,N} \\
| \text{if } k' \neq \bullet \text{ then } s_j := 1 \\
| \text{return } (s_1, \ldots, s_r) \\
\]

We now lower bound the distinguishing advantage \( \Delta^{D_\ell}(R_T, S_T) \). Note that when \( D_\ell \) interacts with \( R_T \), the distinguisher breaks a given session \( j^* \) with probability exactly \( \max_{D \in \mathcal{D}} \Gamma^T(j^*, D) \). If session \( j^* \) is broken, then \( D_\ell \) always outputs 1 by correctness of the encryption scheme. Thus,

\[
P^{DR_T}(1) = \frac{1}{r} \sum_{j^* \in \{1, \ldots, r\}} \max_{D \in \mathcal{D}} \Gamma^T(j^*, D) = \Gamma^T_{\text{opt}}.
\]

In contrast, when the distinguisher \( D_\ell \) interacts with \( S_T \), the event that matters is whether the simulator \( \sigma \) managed to break the session \( j^* \) before sending the ciphertext \( c_{j^*} \), even though the session \( j^* \) to be broken
will only be chosen afterwards by $D_\ell$. In the random experiment $D_\ell S_T$, let $G'_j$ be the event that the session $j$ was broken at least once (the trigger $T$ output at least once a switch value with $s_j = 1$) before the simulator $\sigma$ outputs the simulated ciphertext $c_j$, for all $j \in \{1, \ldots, r\}$.

Note that if the simulator $\sigma$ in $S$ manages to break the chosen session $j^*$ before having sent the ciphertext $c_{j^*}$, i.e., $G'_{j^*} = 1$, then $\sigma$ has the possibility of retrieving the corresponding message $m_{j^*}$ and thus could potentially perfectly simulates this ciphertext, so that $\Pr^{D_\ell S_T}(1 | G'_{j^*} = 1) = 1$ is possible. In contrast, when $G'_{j^*} = 0$, the simulator $\sigma$ is committed to the ciphertext $c_{j^*}$ without knowing anything about the message $m_{j^*}$ but its length $\ell$. For the distinguisher $D_\ell$ to output 1 in this case, the simulator $\sigma$ must be able to provide a key $k'$ such that $\text{dec}(k', c_{j^*}) = m_{j^*}$. Since the decryption algorithm $\text{dec}$ is deterministic, it implies that for a fixed ciphertext $c_{j^*}$ the output of $\text{dec}(\cdot, c_{j^*})$ can take at most $|K|$ values. The message $m_{j^*}$ is uniformly distributed over $M_\ell$ and thus 

$$\Pr^{D_\ell S_T}(1 | G'_{j^*} = 0) \leq \frac{|K|}{|M_\ell|}. $$

Note that if the simulator $\sigma$ can provoke the event $G'_j = 1$ in the $D_\ell S_T$ random experiment, then $\sigma$ can be used to win the event $G_j = 1$ against the trigger $T$ alone. Since $\sigma$ does not know in advance which session is going to be selected by $D_\ell$, its success is therefore at most the maximum of the average, i.e.,

$$\frac{1}{r} \sum_{j^* \in \{1, \ldots, r\}} \Pr^{D_\ell S_T}(G'_{j^*} = 1) \leq \Gamma_{\text{avg}}^T.$$

Thus,

$$\Pr^{D_\ell S_T}(1) = \frac{1}{r} \sum_{j^* \in \{1, \ldots, r\}} \sum_{b \in \{0,1\}} \Pr^{D_\ell S_T}(G'_{j^*} = b) \cdot \Pr^{D_\ell S_T}(1 | G'_{j^*} = b)$$

$$\leq \frac{1}{r} \sum_{j^* \in \{1, \ldots, r\}} \Pr^{D_\ell S_T}(G'_{j^*} = 1) + \frac{|K|}{|M_\ell|}$$

$$\leq \Gamma_{\text{avg}}^T + \frac{|K|}{|M_\ell|}.$$ 

Combining the previous equations gives the desired lower bound. 

### 3.4.3 PBE with Local Assumptions

As mentioned in Remark 3.1, our impossibility result does not apply to the particular case of the local password-guessing trigger $LT(P, q)$ if the
passwords are independently distributed, allowing for the existence of password-based encryption under these assumptions. Intuitively, since each session has its own restriction on the number of password-guessing queries, the simulation strategy can optimally brute-force each session independently to avoid the commitment problem in the same way as in the case of a single session in Section 3.4.1.

We therefore assume in this section that the passwords are independently, but not necessarily identically, distributed. That is, there exist r password distributions $\mathcal{P}_1, \ldots, \mathcal{P}_r$ such that the distribution $\mathcal{P}$ of the r passwords can be written as $\mathcal{P}(w_1, \ldots, w_r) = \mathcal{P}_1(w_1) \times \cdots \times \mathcal{P}_r(w_r)$, for all $w_1, \ldots, w_r \in \mathcal{W}$. Such a distribution $\mathcal{P}$ will be called a product distribution.

Note that passwords in $\mathcal{W}$ can be sorted according to their likelihood of being chosen in the $j$th session as given by $\mathcal{P}_j$. We denote by $\gamma_{j,q} \in [0,1]$ the probability that the password selected in the $j$th session belongs to the $q$ most likely passwords (according to $\mathcal{P}_j$), for all $j \in \{1, \ldots, r\}$ and any integer $q$.

The following lemma shows that when constructing the downgradable secure channel $\text{SEC}_r^{\text{LT}}(\mathcal{P}, q)$ from the downgradable key $\text{KEY}_r^{\text{LT}}(\mathcal{P}, q)$ where the switch value can be changed through the trigger system $\text{LT}(\mathcal{P}, q)$, it is sufficient to instead look at systems $\text{KEY}_r^{S*}$ and $\text{SEC}_r^{S*}$ where the switch value is randomly chosen and fixed at the beginning according to some random variable $S^*$.

**Lemma 3.2.** Let $\mathcal{P}$ be a product distribution of r passwords as described above and consider a tuple of r integers $q := (q_1, \ldots, q_r)$. For each session $j \in \{1, \ldots, r\}$, let $S_j^*$ be a binary random variable which is 1 with probability $\gamma_{j,q_j}$, and let $S^* := (S_1^*, \ldots, S_r^*)$. Then, there exist two converters $\sigma_{\text{KEY}}$ and $\sigma_{\text{SEC}}$ described in Systems 3.10 and 3.11, respectively, such that

$$\text{KEY}_r^{\text{LT}}(\mathcal{P}, q) \equiv \sigma_{\text{KEY}}^E \text{KEY}_r^{S^*} \quad \text{and} \quad \text{SEC}_r^{S^*} \equiv \sigma_{\text{SEC}}^E \text{SEC}_r^{\text{LT}}(\mathcal{P}, q).$$

**Proof.** The core systems $\text{KEY}^r$ and $\text{SEC}^r$ introduced in Section 3.1.2 are such that a change in their switch value modifies the behavior of the system only at the adversarial $E_N$ interface and not at the others (interface A or B).

In the case of the key system, the converter $\sigma_{\text{KEY}}$ has to be indistinguishable from $\text{KEY}_r^{\text{LT}}(\mathcal{P}, q)$, where the value of the switch can be changed progressively, by having only access to $\text{KEY}_r^{S^*}$, where the value of the switch is (randomly) fixed at the beginning. To do so, the converter $\sigma_{\text{KEY}}$ uses the fact it can easily infer the value of $S^*$ by simply querying $(j, \text{getkey})$ at the $E_N$-interface of $\text{KEY}_r^{S*}$. If the answer contains the error symbol $\bullet$, then $S_j^* = 0$, and otherwise $S_j^* = 1$, for all $j \in \{1, \ldots, r\}$. If $S_j^* = 0$,
then $\sigma_{\text{KEY}}$ cannot retrieve the key associated to this session and therefore declares any password guess for that session as invalid. In contrast, if $S_j^* = 1$, which happens with probability $\gamma_{j,q_j}$, the converter $\sigma_{\text{KEY}}$ must then “scale up” the password-guessing probabilities in order to match that of $\text{KEY}_{\text{LT}}(P,q)$. A password guess $w$ for a session where $S_j^* = 1$ is considered valid if a coin tossed with probability $\frac{P_j(w)}{\gamma_{j,q_j}}$ is 1. Note that this is truly a probability distribution since $\gamma_{j,q_j}$ represents the winning probability of the optimal guessing strategy for the $j$th session with $q_j$ queries and therefore $P_j(w) \leq \gamma_{j,q_j}$, for all password guesses $w \in W$. Overall, the probability of retrieving the key used in the $j$th session is therefore $P_j(w)$ and thus $\text{KEY}_{\text{LT}}(P,q) \equiv \sigma_{\text{E KEY}_{S^*}}$. A more formal description of $\sigma_{\text{KEY}}$ is in System 3.10.

**System 3.10: Converter $\sigma_{\text{KEY}}$**

$s_j := 0$ and $k_j := \bullet$, for all $j \in \{1, \ldots, r\}$

**on input** $(j, w)$ at out$_S$

- $(j, k) := \text{result of querying } (j, \text{getkey}) \text{ at in}_N$
  - if $k \neq \bullet$ then
    - flip a bit $b$ with probability $\frac{P_j(w)}{\gamma_{j,q_j}}$
    - $s_j := s_j \lor b$ and $k_j := k$

**on input** $(j, \text{getkey})$ at out$_N$

- if $s_j = 1$ then **output** $(j, k_j)$ at out$_N$
- else **output** $(j, \bullet)$ at out$_N$

In contrast to $\sigma_{\text{KEY}}$, the task of $\sigma_{\text{SEC}}$ is to emulate $\text{SEC}_{S^*}$, where the value of the switch is (randomly) fixed at the beginning, from $\text{SEC}_{\text{LT}}(P,q)$, where its value can be changed progressively. To do so, $\sigma_{\text{SEC}}$ simply does the optimal password-guessing strategy, i.e., $\sigma_{\text{SEC}}$ queries the $q_j$ most likely passwords for the $j$th session, for all $j \in \{1, \ldots, r\}$, and hence $\text{SEC}_{S^*} \equiv \sigma_{\text{E SEC}_{\text{LT}}(P,q)}$. A more detailed description of $\sigma_{\text{SEC}}$ is in System 3.11.

**Remark 3.2.** We point out that the strategies described by the converters $\sigma_{\text{KEY}}$ or $\sigma_{\text{SEC}}$ in Lemma 3.2 have demanding preconditions: They need to know the password distribution and, for each session, to also know an upper bound on the number of password-guessing queries as well as to be able to sort through the set of passwords in order to determine the most likely ones.
3.4 Password-Based Encryption

System 3.11: Converter $\sigma_{\text{SEC}}$

\begin{verbatim}
\begin{align*}
b_{\text{forced}}_j & := 0 \text{ for all } j \in \{1, \ldots, r\} \\
on \text{input} \ (j, |m|) & \text{ at } in_N \\
| \text{if } b_{\text{forced}}_j = 0 \text{ then } & \text{BruteForce}(j, q_j) \\
\text{output} \ (j, |m|) & \text{ at } in_N \\
on \text{input} \ (j, \text{getmsg}) & \text{ at } out_N \\
| \text{if } b_{\text{forced}}_j = 0 \text{ then } & \text{BruteForce}(j, q_j) \\
(j, m') & := \text{result of querying} \ (j, \text{getmsg}) \text{ at } in_N \\
\text{output} \ (j, m') & \text{ at } out_N \\
\end{align*}
\end{verbatim}

Procedure $\text{BruteForce}(j, q)$

\begin{verbatim}
\begin{align*}
& \text{for } 1 \leq k \leq q \text{ do} \\
& \quad w_k := k^{\text{th}} \text{ most likely password according to } \mathcal{P}_j \\
& \quad \text{output} \ (j, w_k) \text{ at } in_S \\
& \quad b_{\text{forced}}_j := 1 \\
& \end{align*}
\end{verbatim}

The next lemma shows that the protocol $(\text{enc}, \text{dec})$ constructs a secure channel $\text{SEC}_S^r$ from a key resource $\text{KEY}_S^r$ and authenticated communication $\text{AUT}^r$ if the underlying encryption scheme is IND-CPA secure, for every random switch value $S$ over \{0, 1\}$^r$.

Lemma 3.3. Let $SE := (\text{enc}, \text{dec})$ be a correct encryption scheme and consider the associated protocol $se := (\text{enc}, \text{dec})$. Let $S$ be a random variable arbitrarily distributed over $\{0, 1\}$\(^r\). Then, there exist a simulator $\sigma_{\text{CPA}}$ described in System 3.12 and an (efficient) reduction system $C_{\text{CPA}}$ described in the proof such that

\[
\begin{array}{cccc}
[\text{KEY}_S^r, \text{AUT}^r] & \xrightarrow{(se, \sigma_{\text{CPA}}, \varepsilon)} & \text{SEC}_S^r,
\end{array}
\]

where $\varepsilon(D) := r \cdot \Delta_{\text{DCPA}}(G_0^{\text{CPA}}(SE), G_1^{\text{CPA}}(SE))$, for all distinguishers $D$.

Proof. Let $S$ be a random switch value arbitrarily distributed over $\{0, 1\}$\(^r\). Although the simulator $\sigma_{\text{CPA}}$ does a priori not know the value of the random switch $S$, it can easily retrieve it by querying the channel $\text{SEC}_S^r$ on $(j, \text{getmsg})$. If the answer contains the error symbol $\Diamond$, then $S_j = 0$, otherwise $S_j = 1$, for all $j \in \{1, \ldots, r\}$. If $S_j = 0$, then the simulator $\sigma_{\text{CPA}}$ will never be able to retrieve the transmitted message of the $j^{\text{th}}$ session and simulates therefore the transmission of a ciphertext by encrypting a random message of the correct length under a random key. For such
a session, any key retrieval query \((j, \text{getkey})\) will be responded with the error symbol \(\blacklozenge\). In the other case, if \(S_j = 1\), the simulator \(\sigma_{\text{CPA}}\) can simply retrieve the transmitted message and encrypt it under a uniform random key to simulate perfectly the sent ciphertext. For such a session, the actual key used for simulating the encryption will be leaked if the simulator \(\sigma_{\text{CPA}}\) receives a key retrieval query. The simulator \(\sigma_{\text{CPA}}\) is described more precisely in System 3.12.

**System 3.12:** Simulator \(\sigma_{\text{CPA}}\)

\[
k_1, \ldots, k_r \leftarrow \{0, 1\}^n
\]

**on input** \((j, |m|)\) at \(\text{in}_N\)

\[
(j, m'_j) := \text{result of querying} \ (j, \text{getmsg}) \text{ at } \text{in}_N
\]

**if** \(m'_j = \blacklozenge\) **then**

\[
m'_j \leftarrow \{0, 1\}^{|m|}
\]

\[
c \leftarrow \text{enc} (k_j, m'_j)
\]

**output** \((j, c)\) at \(\text{out}_2\)

**on input** \((j, \text{getkey})\) at \(\text{out}_{1,N}\)

\[
(j, m'_j) := \text{result of querying} \ (j, \text{getmsg}) \text{ at } \text{in}_N
\]

**if** \(m'_j = \blacklozenge\) **then** \textbf{output} \((j, \blacklozenge)\) at \(\text{out}_{1,N}\)

**else** \textbf{output} \((j, k_j)\) at \(\text{out}_{1,N}\)

In order to have shorter notations within the proof, let us denote by \(\mathbf{R}_S\) and \(\mathbf{S}_S\) the real system \(\text{enc}^A \text{dec}^B [\text{KEY}^*_S, \text{AUT}^*_S]\) and the ideal system \(\sigma^E \text{SEC}^*_S\), respectively. Note that for sessions which are “broken” \((S_j = 1)\) the simulation is perfect, while for the other sessions the only difference between the two systems \(\mathbf{R}_S\) and \(\mathbf{S}_S\) lies in the way ciphertexts are produced: \(\mathbf{R}_S\) produces encryptions of messages which were input at interface \(A\), whereas \(\mathbf{S}_S\) produces encryptions of random messages of the same length as messages input at \(A\). Distinguishing \(\mathbf{R}_S\) from \(\mathbf{S}_S\) can therefore be reduced to the IND-CPA security of the underlying encryption scheme via a hybrid argument as follows.

We define a sequence of reduction systems \(\mathbf{C}^{\text{CPA}}_1, \ldots, \mathbf{C}^{\text{CPA}}_r\), where each reduction \(\mathbf{C}^{\text{CPA}}_{j^*}\) has 4 outside sub-interfaces (labeled \(A, B, E_{1,N}, \text{ and } E_2\) ) and connects at its inside interface in to one of the two IND-CPA systems \(\mathbf{G}^{\text{CPA}}_0 (\text{SE})\) or \(\mathbf{G}^{\text{CPA}}_1 (\text{SE})\) described in System 2.2 in Section 2.6.3, for all \(j^* \in \{1, \ldots, r\}\). Initially, the reduction system \(\mathbf{C}^{\text{CPA}}_{j^*}\) selects \(r\) keys \(k_1, \ldots, k_r\) uniformly at random, as well as a switch value \((s_1, \ldots, s_r)\) according to \(P_S\), the distribution of the random switch \(S\). Upon input \((j, m)\) at its outside
sub-interface A, the reduction system $C_{j^*}^{CPA}$ outputs $(j, m)$ at its outside sub-interface B and $(j, c)$ at $E_2$, where the ciphertext $c$ is computed as follows. If session $j$ is “broken” ($s_j = 1$), then $c$ is a local encryption of the input message $m$ under key $k_j$. Otherwise, if session $j$ is not “broken” ($s_j = 0$), then $c$ is a local encryption of a fresh random message of length $|m|$ under key $k_j$ for session $j < j^*$, while for session $j > j^*$ the ciphertext $c$ is an actual encryption of the input message under key $k_j$. Finally, only for an unbroken session $j = j^*$ is the ciphertext $c$ the result of querying the system connected at its inside interface on the message $m$. Key retrieval queries $(j, \text{getkey})$ made to the outside sub-interface $E_{1, N}$ of $C_{j^*}^{CPA}$ are replied by $(j, \diamond)$ or $(j, k_j)$ according to the value of $s_j$. The reduction system $C_{j^*}^{CPA}$ is described more formally in System 3.13.

**System 3.13: Reduction $C_{j^*}^{CPA}$**

$k_1, \ldots, k_r \leftarrow \{0, 1\}^n$

$(s_1, \ldots, s_r) \leftarrow P_S$

**on input $(j, m)$ at A**

\[
\begin{align*}
  c &:= \diamond \\
  \text{if } s_j = 0 \text{ then} \\
  \quad \text{if } j < j^* \text{ then} \\
  \quad \quad c &\leftarrow \text{enc}(k_j, m_1), \text{ where } m_1 \leftarrow \{0, 1\}^{|m|} \\
  \quad \text{else if } j = j^* \text{ then} \\
  \quad \quad c &:= \text{result of querying } m \text{ at in} \\
  \quad \text{else} \\
  \quad \quad c &\leftarrow \text{enc}(k_j, m) \\
  \text{else } c &\leftarrow \text{enc}(k_j, m) \\
  \text{output } (j, m) \text{ at B} \\
  \text{output } (j, c) \text{ at } E \\
\end{align*}
\]

**on input $(j, \text{getkey})$ at $E_N$**

\[
\begin{align*}
  \text{if } s_j = 0 \text{ then output } (j, \diamond) \text{ at } E_N \\
  \text{else output } (j, k_j) \text{ at } E_N \\
\end{align*}
\]

Note that ciphertexts produce by $C_1^{CPA}G_0^{CPA}$ (SE) are always encryptions of messages which were input at interface A and thus $R_S \equiv C_1^{CPA}G_0^{CPA}$ (SE). Similarly, ciphertexts produce by $C_r^{CPA}G_1^{CPA}$ (SE) are encryptions of random messages of the same length as messages input at A, unless they are for a “broken” session in which case the actual input message is encrypted, and thus $S_S \equiv C_r^{CPA}G_1^{CPA}$ (SE). Furthermore, note that unless a session is broken,
both \( C_{j+1}^{\text{CPA}} G_{0}^{\text{CPA}} (SE) \) and \( C_{j}^{\text{CPA}} G_{1}^{\text{CPA}} (SE) \) produce encryptions of random messages of the appropriate length until the \( j \)th session, while ciphertexts for sessions \( j + 1, \ldots, r \) are actual encryptions of input messages. Thus, the sequence of reduction systems \( C_{1}^{\text{CPA}}, \ldots, C_{r}^{\text{CPA}} \) is such that \( C_{j+1}^{\text{CPA}} G_{0}^{\text{CPA}} (SE) \equiv C_{j}^{\text{CPA}} G_{1}^{\text{CPA}} (SE) \), for all \( j \in \{1, \ldots, r - 1\} \). For a reduction system \( C^{\text{CPA}} \) which selects a session \( j^{*} \in \{1, \ldots, r\} \) uniformly at random and then implements the reduction \( C_{j^{*}}^{\text{CPA}} \) we have

\[
\Delta_{D}^{DC^{\text{CPA}}} (G_{0}^{\text{CPA}} (SE), G_{1}^{\text{CPA}} (SE)) = \frac{1}{r} \cdot \Delta_{D}^{D} (R_{S}, S_{S}),
\]

for every distinguisher \( D \).

The previous lemmas allow us to easily prove the main result of this section, which is that password-based encryption with these additional local assumptions is possible if the encryption scheme used is IND-CPA secure and the \( r \) passwords are independently distributed.

**Theorem 3.3.** Let \( SE := (enc, dec) \) be a correct encryption scheme and consider the associated protocol \( se := (enc, dec) \). For every product distribution \( \mathcal{P} \) of \( r \) passwords and every tuple of \( r \) integers \( q := (q_{1}, \ldots, q_{r}) \), there exist a simulator \( \sigma_{LT} \) and an (efficient) reduction system \( C^{LT} \) described in the proof such that

\[
\left[ KEY_{LT}^{r}(\mathcal{P}, q), AUT^{r} \right] \xrightarrow{\text{(se, } \sigma_{LT}, \varepsilon \text{)}}_{\text{ABE}} \text{SEC}_{LT}^{r}(\mathcal{P}, q),
\]

where \( \varepsilon (D) := r \cdot \Delta_{D}^{DC^{LT}} (G_{0}^{\text{CPA}} (SE), G_{1}^{\text{CPA}} (SE)) \), for every distinguisher \( D \).

**Proof.** Consider a product distribution \( \mathcal{P} \) of \( r \) passwords and a tuple of \( r \) integers \( q := (q_{1}, \ldots, q_{r}) \). In order to have shorter notations within the proof, let \( R_{LT}(\mathcal{P}, q) \) denote the downgradable resource \( \left[ KEY_{LT}^{r}(\mathcal{P}, q), AUT^{r} \right] \) and let \( S_{LT}(\mathcal{P}, q) \) denote \( \text{SEC}_{LT}^{r}(\mathcal{P}, q) \). Given the encryption protocol \( se := (enc, dec) \) and some resource \( U \), it will be convenient to use the shorthand \( se U := enc^{A} \cdot dec^{B} U \).

Then, Lemma 3.2 implies that there exist two converters \( \sigma_{\text{key}} \) and \( \sigma_{\text{sec}} \), as well as a random variable \( S^{*} \) over \( \{0, 1\}^{r} \), such that

\[
R_{LT}(\mathcal{P}, q) \equiv \sigma_{\text{key}}^{E} R_{S^{*}} \quad \text{and} \quad S_{S^{*}} \equiv \sigma_{\text{sec}}^{E} S_{LT}(\mathcal{P}, q),
\]

where \( \sigma_{\text{key}}^{*} := \langle \sigma_{\text{key}}, 1 \rangle \) and accounts for the presence in parallel of the channel \( AUT^{r} \) in \( R \). Lemma 3.3 implies then that for such a random switch...
value $S^*$, there exist a simulator $\sigma_{\text{CPA}}$ and a reduction $C_{\text{CPA}}$, such that for all distinguishers $D$

$$\Delta^D (\text{se} \, R_{S^*}, \sigma_{\text{CPA}}^E S_{S^*}) \leq r \cdot \Delta^{DC_{\text{CPA}}} (G_{0_{\text{CPA}}}^E (\text{SE}), G_{1_{\text{CPA}}}^E (\text{SE})) .$$

The proof then follows by composition. To see that, note that the previous system equation implies that

$$\text{se} \, R_{\text{LT}(P,q)} \equiv \text{se} \, \sigma'_{\text{KEY}} R_{S^*} \equiv \sigma'_{\text{KEY}} \, \text{se} \, R_{S^*},$$

where we used the fact that converters connected at different interfaces commute. Let the simulator $\sigma_{\text{LT}}$ be defined as the sequential composition of the aforementioned simulators, i.e., $\sigma_{\text{LT}} := \sigma'_{\text{KEY}} \circ \sigma_{\text{CPA}} \circ \sigma_{\text{SEC}}$. Then, we have $\sigma_{\text{LT}}^E S_{\text{LT}(P,q)} \equiv \sigma'_{\text{KEY}} \sigma_{\text{CPA}}^E S_{S^*}$, and thus

$$\Delta^D (\text{se} \, R_{\text{LT}(P,q)}, \sigma_{\text{LT}}^E S_{\text{LT}(P,q)}) \leq \Delta^{D'} (\text{se} \, R_{S^*}, \sigma_{\text{CPA}}^E S_{S^*}),$$

where $D' := D \sigma'_{\text{KEY}}^E$. The proof is then finished by combining both distinguishing advantages and defining the overall reduction $C_{\text{LT}}$ as $C_{\text{LT}} := \sigma'_{\text{KEY}}^E C_{\text{CPA}}$.

**Remark 3.3.** The simulation strategy employed by $\sigma_{\text{LT}}$ in the construction of Theorem 3.3 is rather peculiar. Informally stated, whenever $\sigma_{\text{LT}}$ has to simulate a ciphertext for a particular session, it first tries to brute force the password of this session by probing the most likely passwords (this step is taken care of by $\sigma_{\text{SEC}}$ in System 3.11). If this step succeeds, $\sigma_{\text{LT}}$ can perfectly simulate the transmitted ciphertext, otherwise it simply encrypts a random message of the appropriate length (this step is taken care of by $\sigma_{\text{CPA}}$ in System 3.12). Finally, even though $\sigma_{\text{LT}}$ exhausts its password-guessing queries at the beginning, it can emulate the correct answer to an outside password-guessing query by appropriately scaling the probabilities (this step is taken care of by $\sigma_{\text{KEY}}$ in System 3.10). Note that the assumptions discussed in Remark 3.2 are also required for $\sigma_{\text{LT}}$.

### 3.4.4 Salting and Per-Session Security

In this section we examine the well-known *salting technique*, a standard tool to achieve domain separation in password hashing. This technique consists of prefixing all queries made to a single random oracle by a distinct bit string in each of the $r$ sessions, making the queries from different sessions land in different subdomains of the random oracle. In practice, a randomly chosen bit string is used for every session, maintaining the same properties with high probability.
Assumed resources. The construction statement below assumes the availability of a single random oracle and the strings used for salting. The random oracle $\mathsf{RO}_q$ is restricted by allowing Eve to ask at most $q$ queries. The source of salts is formalized as a resource $\mathsf{SALT}$ and parameterized by a distribution $\mathcal{R}$ of $r$ distinct $t$-bit strings, for some appropriate integer $t$. The resource $\mathsf{SALT}(\mathcal{R})$ first samples $(v_1, \ldots, v_r)$ from $\mathcal{R}$ and then outputs $(j, v_j)$ at interface $i \in \{A, B, E\}$ whenever it receives a query $(j, \mathsf{getsalt})$ at the same interface, for any $j \in \{1, \ldots, r\}$. Note that the prefixes used in each session are public and can be retrieved by Eve.

The prefixing protocol. Both Alice and Bob use the bit strings provided by the salting resource to prefix their queries as described by the following converter $\mathsf{pre}$: Upon input $(j, x)$ at its interface $\mathsf{out}$, corresponding to hashing a message $x \in \{0, 1\}^*$ for the $j^{th}$ session, the converter $\mathsf{pre}$ retrieves the prefix $v_j$ associated with that session by querying $\mathsf{SALT}(\mathcal{R})$ on $(j, \mathsf{getsalt})$ and then outputs the value returned by the single random oracle when queried on $v_j \parallel x$.

Desired resource. The goal of the prefixing protocol $\mathsf{pre} := (\mathsf{pre}, \mathsf{pre})$ is to construct $r$ independent random oracles. Since the assumed random oracle $\mathsf{RO}_q$ can be queried by Eve at most $q$ times, the constructed resource will naturally have a similar restriction. Namely, the following lemma shows that the prefixing protocol $\mathsf{pre}$ constructs $r$ globally restricted random oracles $[\mathsf{RO}, \ldots, \mathsf{RO}]_q$.

**Lemma 3.4.** Consider the prefixing protocol $\mathsf{pre} := (\mathsf{pre}, \mathsf{pre})$ described above. Then, for every distribution $\mathcal{R}$ of $r$ distinct bit strings of equal length and every integer $q$, there exists a simulator $\sigma_{\mathsf{pre}}$ such that

$$[\mathsf{RO}_q, \mathsf{SALT}(\mathcal{R})] \xrightarrow{\mathsf{pre}, \sigma_{\mathsf{pre}}, 0}_{\mathsf{ABE}} [\mathsf{RO}, \ldots, \mathsf{RO}]_q .$$

**Proof.** The simulator $\sigma_{\mathsf{pre}}$ first selects $r$ bit strings $(v_1, \ldots, v_r)$ of equal length according to the distribution $\mathcal{R}$. Then, whenever a salt retrieval query $(j, \mathsf{getsalt})$ for the $j^{th}$ session is input at its interface $\mathsf{out}_2$, the simulator $\sigma_{\mathsf{pre}}$ returns $(j, v_j)$ at the same interface. When a message $x \in \{0, 1\}^*$ to be hashed is input at its interface $\mathsf{out}_1$, the simulator $\sigma_{\mathsf{pre}}$ first checks whether one of the strings $v_j$ is a prefix of $x$, i.e., whether $x = v_j \parallel x'$ for some $j \in \{1, \ldots, r\}$ and $x' \in \{0, 1\}^*$. If this is the case, then $\sigma_{\mathsf{pre}}$ returns the answer of the $j^{th}$ random oracle when queried on $x'$. Otherwise, the simulator $\sigma_{\mathsf{pre}}$ simply returns a uniform $n$-bit string. The simulator $\sigma_{\mathsf{pre}}$ replies consistently to any repeating query and in addition keeps track of
the number of queries to avoid too many “dummy” queries, i.e., queries which are not prefixed by one of the strings $v_j$. It can be readily verified that the simulation is perfect. \hfill \Box

Unfortunately, it is easy to show that the same salting protocol $\text{pre}$ cannot construct $r$ locally restricted random oracles $[\text{RO}_{q_1}, \ldots, \text{RO}_{q_r}]$, at least not unless $q_j \geq q$ for all $j \in \{1, \ldots, r\}$ (the latter would render this construction uninteresting due to the blow-up in the number of adversarial queries). To see this, consider the systems $R := \text{pre}_A \text{pre}_B [\text{RO}_q, \text{SALT}(R)]$ and $S := \sigma^E [\text{RO}_{q_1}, \ldots, \text{RO}_{q_r}]$ for some arbitrary simulator $\sigma$. In the real experiment, a distinguisher making a query $(j, x)$ to the interface $A$ and a query $(j, v_j \| x)$ to the interface $E_1$ of $R$, where $v_j$ is the prefix for the $j$th session output by the salting resource, observes the same output. Hence, in the ideal experiment, for any query of the type $(j, v_j \| x)$ the simulator $\sigma$ has to query the $j$th random oracle $\text{RO}_{q_j}$ on $x$ in order to be able to mimic this behavior. Thus, a distinguisher can choose any session $j$ and make all its $q$ queries to the associated random oracle $\text{RO}_{q_j}$ by appropriately prefixing each query, requiring $q_j \geq q$ for any successful simulator to exist.

**Consequences for local restrictions.** The above observation implies that relying on local query restrictions for multi-session security of password-based encryption (as in Theorem 3.3 and [BRT12]) appears to be in general rather unrealistic. The salting technique employed in the PKCS #5 standard [Kal00] (and more generally, any domain separation technique which is public) fails to construct multiple locally restricted random oracles $[\text{RO}_{q_1}, \ldots, \text{RO}_{q_r}]$ from a single random oracle $\text{RO}_q$ for any meaningful values of $q_1, \ldots, q_r$.

### 3.4.5 On the Per-Session Security of PBE from PKCS #5

In this section we show how the arguments used to prove Theorem 3.2 in Section 3.4.2 also apply to the password-based encryption standard described in PKCS #5 [Kal00], which thus in general does not achieve per-session confidentiality.

Recall that the protocol described in [Kal00] consists of hashing a salt concatenated with a password and of using the result as a key for encryption.\footnote{Additionally, it also increases the cost of a brute-force attack by iterative hashing. This is however an orthogonal issue, see Chapter 5 for a detailed discussion.} Formalized as a pair of converters $(\text{pbe}, \text{pbd})$, this protocol
corresponds to first doing the salting step via the protocol \((\text{pre, pre})\) described in Section 3.4.4, then the key derivation protocol \((\text{kd, kd})\) described in Section 3.2 and finally symmetric encryption \((\text{enc, dec})\) described in Section 3.4. As detailed in previous sections, such a protocol assumes the following resources to be available: a random oracle \(\text{RO}_q\), a source of salts \(\text{SALT}(\mathcal{R})\), a source of shared passwords \(\text{PW}(\mathcal{P})\), and finally an authenticated communication channel \(\text{AUT}^r\) for each of the \(r\) sessions. We denote by \(\text{RKDF}(q, \mathcal{R}, \mathcal{P})\) the resources used for key derivation, i.e.,

\[
\text{RKDF}(q, \mathcal{R}, \mathcal{P}) := [\text{RO}_q, \text{SALT}(\mathcal{R}), \text{PW}(\mathcal{P})].
\]

**Theorem 3.4.** Let \(\text{SE} := (\text{enc, dec})\) be a correct encryption scheme with key space \(\mathcal{K} := \{0, 1\}^n\) and message space \(\mathcal{M} \subseteq \{0, 1\}^*\), and consider the combined protocol \((\text{pbe, pbd})\) described above. Let \(\mathcal{T}\) be a trigger system with input space \(\{1, \ldots, r\} \times \mathcal{W}\) and output space \(\{0, 1\}^r\), and let \(\mathcal{M}_\ell\) denote a non-empty set of messages of fixed length \(\ell\) in \(\mathcal{M}\), for some integer \(\ell\). Then, for every integer \(q\), every distribution \(\mathcal{R}\) of \(r\) distinct bit strings of the same length and every distribution \(\mathcal{P}\) of \(r\) passwords, there exists a distinguisher \(D'_\ell\) described in System 3.14 such that for all simulators \(\sigma\) we have

\[
\Delta_{D'_\ell} \left( \text{pbe}^A \text{pbd}^B \left[ \text{RKDF}(q, \mathcal{R}, \mathcal{P}), \text{AUT}^r \right], \sigma^E \text{SEC}^r \right) \geq \delta_{\mathcal{T}}^{\text{GT}(\mathcal{P}, q)} - \mu,
\]

where \(\delta_{\mathcal{T}}^{T_2} := \Gamma_{\text{opt}}^{T_2} - \Gamma_{\text{avg}}^{T_1}\), with \(\Gamma_{\text{opt}}^{T_2}\) and \(\Gamma_{\text{avg}}^{T_1}\) defined in (3.1), and

\[
\mu \leq r \cdot \Delta_{D'_{\ell, q}}^{\text{CPA}} \left( \mathcal{G}_0^{\text{CPA}}(\text{SE}), \mathcal{G}_1^{\text{CPA}}(\text{SE}) \right) + (r + 1) \cdot \frac{|\mathcal{K}|}{|\mathcal{M}_\ell|},
\]

for a distinguisher \(D_{\ell, q}^{\text{CPA}}\) described in System 3.15.

The proof is analogous to that of Theorem 3.2 and consequently the same limitations mentioned in Remark 3.1 also apply to Theorem 3.4. Indeed, if the encryption scheme used is IND-CPA secure and uses significantly less keys than messages, then the dominant term in the lower bound above is \(\delta_{\mathcal{T}}^{\text{GT}(\mathcal{P}, q)}\) which may become moot depending on the trigger \(\mathcal{T}\) considered in the constructed resource. However, in the case where the passwords are independently distributed and \(\mathcal{T} := \text{GT}(\mathcal{P}, p)\), the quantity \(\delta_{\mathcal{T}}^{\text{GT}(\mathcal{P}, q)}\) will be large for most password distributions until \(p < rq\), since following the optimal guessing strategy for a particular session usually requires to exhaust all \(q\) guesses. Likewise, if instead \(\mathcal{T} := \text{LT}(\mathcal{P}, q)\), for some integers \(q := (q_1, \ldots, q_r)\), then \(\delta_{\mathcal{T}}^{\text{GT}(\mathcal{P}, q)}\) will be large if \(q_j < q\) for some \(j \in \{1, \ldots, r\}\).
3.4 Password-Based Encryption

**Proof (of Theorem 3.4).** The main idea is to use the same strategy as the distinguisher $D_\ell$ in System 3.9 which, very roughly, consists of the following three steps: 1) input $r$ messages $m_1, \ldots, m_r$ of length $\ell$ chosen uniformly at random; 2) observe the corresponding ciphertexts $c_1, \ldots, c_r$; and 3) “run” the optimal strategy to break a session $j^*$ chosen uniformly at random. We need to adapt this last step since the resources considered, contrary to those in Theorem 3.2, do not necessarily have a switch value $s_j$ indicating whether session $j$ is broken.

In order to have shorter notations within the proof, the real system $\text{pbe}^A \text{pbd}^B [\text{RKDF} (q, R, P), \text{AUT}^r]$ will be denoted by $R$ and the ideal system $\sigma^E \text{SEC}^r$ will be denoted by $S_T$, for some simulator $\sigma$ and some $((1, \ldots, r) \times W, \{0, 1\}^r)$-trigger system $T$. Eve’s interface in $R$ is split as follows: sub-interfaces $E_{1,1}, E_{1,2}$ and $E_{1,3}$ denote the respective $E$-interface of the resources $\text{RO}_q$, $\text{SALT}(R)$ and $\text{PW}(P)$ in the combined resource $\text{RKDF} (q, R, P)$; while the sub-interface $E_2$ of $R$ denotes the $E$-interface of the authenticated channel $\text{AUT}^r$.

Similarly to $D_\ell$, the new distinguisher $D'_\ell$ “runs” the optimal strategy $D_{\text{opt}} (j^*)$ for guessing the password used in a randomly chosen session $j^*$, where

$$D_{\text{opt}} (j) := \arg \max_{D \in \mathcal{D}} \Gamma^{GT(P,q)}(j, D),$$

for all $j \in \{1, \ldots, r\}$. Note that $D_{\text{opt}} (j^*)$ expects as input an $r$-bit string indicating the “broken” state of each session. To do so, the distinguisher $D'_\ell$ keeps a state $(s_1, \ldots, s_r)$ which is updated as follows. Given a password guess $(j, w)$ for session $j$ (which could a priori be different from $j^*$), the new distinguisher $D'_\ell$ declares session $j$ “broken” if the output $k$ of the random oracle when queried on $v_j \parallel w$, where $v_j$ is the prefix used for session $j$, is such that $\text{dec}(k, c_j) = m_j$. The distinguisher $D'_\ell$ is given in more details in System 3.14.

We now lower bound the distinguishing advantage $\Delta^{D'_\ell} (R, S_T)$. Let $G_j$ be the event that the password used in session $j$ was correctly guessed in the $D'_\ell R$ random experiment, i.e., a query prefixed by $v_j$ made to the random oracle at interface $E$ matches a previous query made to the random oracle at interface $A$ or $B$, where $v_j$ is the prefix used for session $j$ output by the salting resource $\text{SALT}(R)$. If the distinguisher $D'_\ell$ manages to guess the password of a given session $j^*$, it will necessarily output 1 due to the correctness of the encryption scheme, i.e.,

$$P^{D'_\ell R}(1) \geq \frac{1}{r} \sum_{j^* \in \{1, \ldots, r\}} P^{D'_\ell R}(G_{j^*} = 1).$$
System 3.14: Distinguisher $\mathbf{D}_\ell'$

\[
m_j \leftarrow \mathcal{M}_\ell, \quad c_j := \diamond \text{ and } s_j := 0, \text{ for all } j \{1, \ldots, r\}
\]

\[
j^* := \diamond
\]

for \(1 \leq j \leq r\) do

\[
\begin{align*}
(j, c_j) & := \text{result at } E_2 \text{ of querying } (j, m_j) \text{ at } A \\
j^* & \leftarrow \{1, \ldots, r\}
\end{align*}
\]

return BreakSession($j^*$)

---

Procedure BreakSession($j$)

\[
(j', w) := 1^{\text{st}} \text{ output of } \mathbf{D}_{\text{opt}}(j)
\]

while \((j', w) \neq \bullet\) do

\[
\begin{align*}
& v_{j'} := \text{result of querying} \\
& (j', \text{getsalt}) \text{ at } E_{1,2} \\
& k_{j'} := \text{result of querying} \\
& v_{j'} \parallel w \text{ at } E_{1,1}
\end{align*}
\]

if \(\text{dec}(k_{j'}, c_{j'}) = m_{j'}\) then

\[
s_{j'} := 1
\]

if \(s_{j'} = 1\) then return 1

\[
t := \text{emulate result of querying } \mathbf{D}_{\text{opt}}(j) \text{ on } (s_1, \ldots, s_r)
\]

return 0

---

However, note that the switch values \((s_1, \ldots, s_r)\) emulated by the distinguisher $\mathbf{D}_\ell'$ and which are input to $\mathbf{D}_{\text{opt}}(j^*)$ have a different distribution than when $\mathbf{D}_{\text{opt}}(j^*)$ interacts with the trigger $\text{GT}(\mathcal{P}, q)$ alone. Indeed, in the $\mathbf{D}_\ell'\mathcal{R}$ random experiment, if a session $j$ is declared to be “broken” ($s_j = 1$), it does not necessarily mean that $\mathbf{D}_{\text{opt}}(j^*)$ correctly guessed the password used in session $j$, it could also be due to either collisions in the random oracle or to the existence of a second key $k$ such that $\text{dec}(k, c_j) = m_j$. This implies that $\mathcal{P}^{\mathbf{D}_\ell'\mathcal{R}}(\mathcal{G}_{j^* = 1})$ may be smaller than $\max_{\mathbf{D} \in \mathcal{D}} \Gamma^{\text{GT}(\mathcal{P}, q)}(j^*, \mathbf{D})$ and we must thus bound this difference.

To do so, let $\mathcal{B}_j$ be the event in the $\mathbf{D}_\ell'\mathcal{R}$ random experiment that a query made to the random oracle at its $E$ interface resulted in session $j$ being declared “broken” ($s_j = 1$) even though the password used in session $j$ was not yet guessed ($\mathcal{G}_j = 0$) and let $\mathcal{B}$ be the union of these events $\mathcal{B}_j$, for all $j \in \{1, \ldots, r\}$. Conditioned on the event $\mathcal{B}$ not happening, a session is declared “broken” if and only if its password is guessed and thus

\[
\mathcal{P}^{\mathbf{D}_\ell'\mathcal{R}}(\mathcal{G}_{j^* = 1}) \geq \max_{\mathbf{D} \in \mathcal{D}} \Gamma^{\text{GT}(\mathcal{P}, q)}(j^*, \mathbf{D}) - \mathcal{P}^{\mathbf{D}_\ell'\mathcal{R}}(\mathcal{B} = 1),
\]

for every session $j^* \in \{1, \ldots, r\}$.

The union bounds implies $\mathcal{P}^{\mathbf{D}_\ell'\mathcal{R}}(\mathcal{B} = 1) \leq \sum_{j \in \{1, \ldots, r\}} \mathcal{P}^{\mathbf{D}_\ell'\mathcal{R}}(\mathcal{B}_j = 1)$ and we now upper bound the probability of the event $\mathcal{B}_j$ happening in the $\mathbf{D}_\ell'\mathcal{R}$ random experiment, for some session $j$. Consider a query $v_j \parallel w$ made to the random oracle at its interface $E$, for some password guess...
3.4 Password-Based Encryption

where $v_j$ is the prefixed associated with session $j$, such that this query provoked the event $s_j = 1$ even though $G_j = 0$. Note that such a query $v_j \parallel w$ was never asked before to the random oracle at interface $E$ (since it provoked the event $s_j = 1$), and was also never queried by the honest parties at interface $A$ or $B$ of the random oracle since the password for that session is not guessed ($G_j = 0$) and other sessions use different prefixes. Thus, the answer of the random oracle on such a fresh query $v_j \parallel w$ is uniformly and independently distributed and conditioned on a given message $m_j$ and ciphertext $c_j$ the probability that $B_j = 1$ is thus at most $q \cdot 2^{-n} |\{k' \in \mathcal{K} \mid \text{dec} (k', c_j) = m_j\}|$ since there are at most $q$ queries made to the random oracle at interface $E$. As $c_j$ is an encryption of $m_j$ under a random key, it then follows that

$$p^{D'_R} (B_j = 1) \leq q \cdot \frac{\kappa_\ell}{2^n},$$

where $\kappa_\ell$ denotes the average number of keys decrypting a ciphertext to the original random message of length $\ell$. That is,

$$\kappa_\ell := \mathbb{E}_{m,c} [\{k' \in \mathcal{K} \mid \text{dec} (k', c) = m\}],$$

where the expectation is taken over a message $m$ chosen uniformly at random in $\mathcal{M}_\ell$ and $c \leftarrow \text{enc} (k, m)$ for a uniformly chosen key $k$.

Note that if the encryption scheme used is IND-CPA secure, then the ratio $q \frac{\kappa_\ell}{2^n}$ will be small. To see that, consider a distinguisher $D^{\text{CPA}}_{\ell,q}$ which interacts with the CPA systems $G^{\text{CPA}}_{b}(SE)$ described in System 2.2. The distinguisher $D^{\text{CPA}}_{\ell,q}$ first selects a message $m$ uniformly at random in $\mathcal{M}_\ell$, then outputs it to $G^{\text{CPA}}_{b}(SE)$ to retrieve the ciphertext $c$ and finally tries to decrypt it by selecting $q$ keys uniformly at random.

**System 3.15:** Distinguisher $D^{\text{CPA}}_{\ell,q}$

$m \leftarrow^{S} \mathcal{M}_\ell$

c := result of querying $m$ to $G^{\text{CPA}}_{b}(SE)$

for $1 \leq u \leq q$ do

$k' \leftarrow^{S} \mathcal{K}$

if $\text{dec} (k', c) = m$ then return 0

return 1

The distinguisher $D^{\text{CPA}}_{\ell,q}$ outputs 0 when connected to $G^{\text{CPA}}_{0}(SE)$ with probability exactly $q \frac{\kappa_\ell}{2^n}$. In contrast, when interacting with $G^{\text{CPA}}_{1}(SE)$, the
ciphertext $c$ produced is independent of the message and thus $D_{\ell, q}^{\text{CPA}}$ outputs 0 with probability at most $\frac{|K|}{|M_\ell|}$. Thus,

$$q \cdot \frac{\kappa_\ell}{2^n} \leq \Delta^D_{\ell, q} \left( G_0^{\text{CPA}} (\text{SE}), G_1^{\text{CPA}} (\text{SE}) \right) + \frac{|K|}{|M_\ell|}.$$  

The previous equations imply that

$$P_{D_R'}(1) \geq \Gamma_{\text{opt}}^{G_T(G, q)} - r \cdot \left( \Delta^D_{\ell, q} \left( G_0^{\text{CPA}} (\text{SE}), G_1^{\text{CPA}} (\text{SE}) \right) + \frac{|K|}{|M_\ell|} \right).$$

Similarly to the proof of Theorem 3.2, the distinguisher $D_R'$ outputs 1 when interacting with the ideal system $S_T$ with probability at most $\Gamma_{\text{avg}}^{G_T(G, q)} + \frac{|K|}{|M_\ell|}$. The desired bound is then obtained by combining the last two equations.

3.5 Conclusion of this Chapter

The work of Bellare et al. [BRT12] initiated the provable-security analysis of the techniques used in the password-based cryptography standard [Kal00], leaving, however, several questions unanswered. First, the different security notions introduced do not formalize a desired per-session type of security guarantee and lack a composition theorem that would make them useful in applications and higher-level protocols. Second, the security statements made are with respect to a class of adversaries having a pre-determined number of password-guessing attempts for each session, which turns out to be a severe restriction even when considering only the simpler case of passwords that are identically and independently distributed. Our formalization overcomes these restrictions.

Even though Theorem 3.3 shows that the results of [BRT12] carry over to a composable model with the corresponding per-session assumptions, the simulation strategy we use is already quite peculiar: the simulator needs to know the password distribution and it must also make all password-guessing attempts before simulating the first ciphertext. This means that the constructed resource allows the attacker to aggregate its entire “computational power” and spend it in advance rather than distributed over the complete duration of the resource use, which results in a weaker guarantee than one might expect.

Finally, our general impossibility result in Theorem 3.2 shows that bounding the adversary’s queries per session, although an unrealistic assumption (as discussed in Section 3.4.4), is necessary for a simulation-based
proof of security of PBE. Otherwise, a commitment problem akin to the one in adaptively secure public-key encryption (PKE) surfaces. Does that mean that we should stop using PBE in practice? In line with Damgård’s [Dam07] perspective on adaptively secure PKE, we view this question as being a fundamental research question still to be answered. On the one hand, we lack an attack that would convincingly break PBE, but on the other hand we also lack provable-security support, to the extent that we can even show the impossibility in our model. Applications using these schemes should therefore be aware of the potential risk associated with their use. We believe that pointing out this commitment problem for PBE, analogously to adaptively secure PKE, is an important contribution of this chapter.
Chapter 4

Query-Restricted Systems

In this chapter, we introduce in Section 4.1 families of constructions parameterized by the number of queries that can be made to the resources. Towards proving such parameterized constructions, Section 4.2 studies further properties of conditional equivalence and provides, as a result of independent interest, a separation between conditional equivalence and the optimality of non-adaptive strategies. Finally, Section 4.3 shows how conditional equivalence is a sufficient criterion to prove the kind of parameterized constructions we aim for.

The results of this chapter will most notably be used in Chapter 5 to deal with query-complexity amplification. Consequently, the findings of this chapter are presented in the same setting as in Chapter 5, the indifferentiability setting described in Section 2.5.2. In particular, resources in this chapter will therefore have two interfaces, a left- and a right-interface, and the set $\mathcal{I}$ of interfaces is $\mathcal{I} := \{\text{left}, \text{right}\}$.

4.1 Parameterized Constructions

Query-restricted systems. We are interested in two-interface systems that only allow a certain number of queries that can be made to their left- or right-interface. This is formalized by extending the considered $(\mathcal{I} \times \mathcal{X}, \mathcal{Y})$-system $\mathbf{S}$ with a monotone binary output (MBO) that captures when the system is exhausted. Notationally, for some integer $q$, we denote by $\mathbf{S}^{[q]}$ the system $\mathbf{S}$ with an MBO that becomes 1 as soon as more than $q$ queries have been made in total to $\mathbf{S}$. We will also consider per-interface
restriction, where for some integers $L, R \in \mathbb{N}$, we denote by $S^{LR}$ the system $S$ with an MBO that becomes 1 as soon as more than $R$ queries have been made at the right-interface of the system $S$, and similarly $LS$ denotes the system $S$ with an MBO that becomes 1 as soon as more than $L$ queries have been made at the left-interface of $S$. If a system has both types of restrictions, we consider the MBO which is the disjunction of the two individual MBOs described above, i.e., $LS^{LR}$ denotes the restricted system allowing at most $L$ queries at the left-interface and at most $R$ queries at the right-interface. We use the same notation for restricting the number of queries at the outside interface of a converter (i.e., we write $L\alpha$ for $\alpha \in \Sigma$ and $L \in \mathbb{N}$), and it is easy to see that for a converter $\alpha$ and a system $S$ we have $L(\alpha S) \equiv (L\alpha)S$ and hence we typically drop the parentheses.

Parameterized families of construction statements. Since we consider only restricted systems throughout this chapter (as well as in Chapter 5), the construction notion for the indistinguishability setting defined in Definition 2.6 will be taken with respect to the restricted distinguishing advantage defined in Section 2.4.3. In the restricted distinguishing advantage $\tilde{\Delta}^D(\cdot, \cdot)$ that we consider, the outputs of a system are blocked once the MBO of the system becomes 1. In the particular case of query-restricted systems this means that the distinguisher $D$ does not obtain further outputs from the system once the specified number of queries is exhausted.

We extend the “arrow notation” from Definition 2.6 to the case where we consider parameterized families of construction statements, where we require that all of the individual statements must hold. More formally, given a space $K$ of parameters and a family of functions $\varepsilon := \{\varepsilon_k\}_{k \in K}$ where each $\varepsilon_k$ is a function mapping each distinguisher to a real number in $[0, 1]$, a family of protocols $\pi := \{\pi_k\}_{k \in K}$ constructs a family of restricted resources $\{S_k\}_{k \in K}$ from an assumed family of restricted resources $\{R_k\}_{k \in K}$, relative to a family of simulators $\sigma := \{\sigma_k\}_{k \in K}$ and within $\varepsilon$, denoted $\{R_k\}_{k \in K}^\text{IND} \rightarrow \{S_k\}_{k \in K}$, if

$$R_k \xrightarrow{\text{IND}} S_k \quad \text{for all } k \in K.$$ 

Typically, the parameter $k$ will be a pair $(L, R)$ indicating the number of queries that can be made to the left- and to the right-interface of the assumed resource. That is, the family of parameterized constructions we
4.2 CE Versus NA

Consider will be of the following format

\[
\left\{ \left( L^R, R^L \right) \right\}_{L,R \in \mathbb{N}} \overset{(\pi, \sigma, \varepsilon)}{\xrightarrow{\text{IND}}} \left\{ S^L \right\}_{L,R \in \mathbb{N}},
\]

where the translation in the number of queries is given by \((\ell, \tau) := \varphi(L, R)\) for some function \(\varphi : \mathbb{N}^2 \to \mathbb{N}^2\), for all \(L, R \in \mathbb{N}\).

**Uniform protocols.** A family of converters \(\alpha = \{\alpha_k\}_{k \in \mathcal{K}}\) is said to be uniform if all the converters in the family are identical without their MBO, i.e., \(\alpha_k^- = \alpha_{k'}^-,\) for all \(k, k' \in \mathcal{K}\). Thus, in a uniform parameterized family of converters, the parameter can only influence the MBO of each converter in the family and can therefore only influence the end of a random experiment (and not the values of the random variables). The reason to consider uniform families of converters is that (semantically) a protocol shall not depend on the number of queries that are made to it, since the restriction is a parameter of the environment in which the protocol is used (and not of the protocol itself). We often denote uniform families of converters only by a symbol that denotes a single converter which has no specified MBO, with the implicit understanding that for each single instance of the construction statement, the converter is amended by an MBO that formalizes the suitable restriction of queries.

A family of constructions will be said to be uniform if both families of protocols and of simulators are uniform. Most families of constructions in this thesis will be uniform and derived via conditional equivalence as described in Section 4.3. Before that, we first describe the implications of conditional equivalence for query-restricted systems.

### 4.2 CE Versus NA

In this section we study several properties of conditional equivalence when dealing with query-restricted systems. The tools developed in this section will be particularly relevant in Section 4.3 to derive parameterized constructions.

**Section outline.** First, we show in Section 4.2.1 that conditional equivalence, similarly to Lemma 2.6, allows to upper bound the restricted distinguishing advantage between (query-) restricted systems by the probability of winning a restricted game in a non-adaptive manner. As a result of independent interest, we also provide a separation result between the
optimality of non-adaptive strategies from conditional equivalence by the following two results about the disjunction of games:

- We show in Section 4.2.2 that if two games are each conditionally equivalent to some system, then the disjunction of both games is conditionally equivalent to the parallel composition of the two other systems.

- In contrast, we show in Section 4.2.3 that adaptivity can help for winning the disjunction of two games, even if the optimal strategies for winning each game were non-adaptive.

Notation. Throughout this section we will use two predicates \( \text{NA}(\mathcal{R}) \) and \( \text{CE}(\mathcal{R}) \) for a game \( \mathcal{R} \), where \( \text{NA}(\mathcal{R}) \) indicates that the best strategy for winning the game \( \mathcal{R} \) is non-adaptive for every fixed number of queries, while \( \text{CE}(\mathcal{R}) \) indicates that \( \mathcal{R} \) is conditionally equivalent to some other system \( \mathcal{S} \).

Definition 4.1. For any \((\mathcal{I} \times \mathcal{X}, \mathcal{Y})\)-game \( \mathcal{R} \), let \( \text{NA}(\mathcal{R}) \) and \( \text{CE}(\mathcal{R}) \) be the following predicates,

\[
\text{NA}(\mathcal{R}) = 1 \iff \forall q \in \mathbb{N} : \hat{\Gamma}(\mathcal{R}_{[q]}) = \hat{\Gamma}^{\text{NA}}(\mathcal{R}_{[q]}),
\]

\[
\text{CE}(\mathcal{R}) = 1 \iff \exists (\mathcal{I} \times \mathcal{X}, \mathcal{Y})\text{-system } \mathcal{S} : \mathcal{R} \equiv \mathcal{S}.
\]

4.2.1 Distinguishing Restricted Systems

We show in the next theorem how the notion of conditional equivalence helps upper-bounding the distinguishing advantage between restricted systems. Recall the non-adaptive distinguisher \([\text{DS}]\) which was shown in Figure 2.1 in Section 2.4.2.

Theorem 4.1. Consider an \((\mathcal{I} \times \mathcal{X}, \mathcal{Y})\)-game \( \mathcal{R} \) and an \((\mathcal{I} \times \mathcal{X}, \mathcal{Y})\)-system \( \mathcal{S} \) such that \( \mathcal{R} \equiv \mathcal{S} \). Then, for any MBO \( A_1, A_2, \ldots \) which is a deterministic function of the inputs and any distinguisher \( \mathcal{D} \),

\[
\hat{\Delta}^D(\mathcal{R}^{[A]}, \mathcal{S}^{[A]}) \leq \hat{\Gamma}^D(\mathcal{R}^{[A]}) = \hat{\Gamma}^D[\mathcal{D}[\mathcal{S}^{[A]^{-1}}](\mathcal{R}^{[A]}).
\]

In particular,

\[
\hat{\Delta}(\mathcal{R}^{[A]}, \mathcal{S}^{[A]}) \leq \hat{\Gamma}^{\text{NA}}(\mathcal{R}^{[A]}).
\]
Proof. The proof basically follows by seeing the MBO indicating the restriction on the number of queries as part of the system’s output and by applying Lemma 2.6. More precisely, let us consider an arbitrary distinguisher $D$ for $(I \times \mathcal{X}, \mathcal{Y} \cup \{\bullet\})$-systems trying to tell apart $[\mathbf{R}^{|A|}]^i$ from $[\mathbf{S}^{|A|}]^i$. Let $D'$ be a distinguisher for $(I \times \mathcal{X}, \mathcal{Y} \times \{0, 1\})$-systems that works as follows: it emulates $D$ but whenever it receives a response $(y, a) \in \mathcal{Y} \times \{0, 1\}$ to any of its queries it forwards instead $y'$ to the distinguisher $D$, where $y' = y$ if $a = 0$, and otherwise $y' = \bullet$. Once the MBO is 1, i.e., $a = 1$ and $D'$ thus forwarded the default value $\bullet$ to $D$, the distinguisher $D'$ does not query further the system it was interacting with, but instead replies to any additional query made by $D$ directly by the dummy symbol $\bullet$. By definition of the distinguisher $D'$ and of the restricted distinguishing advantage $\hat{\Delta}^D$ in Section 2.4.3 it follows that

$$
\hat{\Delta}^D \left( \mathbf{R}^{|A|}, \mathbf{S}^{|A|} \right) = \Delta^{D'} \left( \mathbf{R}^{|A|}, \mathbf{S}^{|A|} \right).
$$

Since the restriction MBO $A_1, A_2, \ldots$ is a deterministic function of the inputs, it follows that given a sequence of inputs the value of the MBO $A$ will be the same in $\mathbf{R}^{|A|}$ and $\mathbf{S}^{|A|}$. Therefore, as the game $\mathbf{R}$ is assumed to be conditionally equivalent to the system $\mathbf{S}$, the game $\mathbf{R}^{|A|}$, where the restriction MBO is seen as part of the output of $\mathbf{R}$, is conditionally equivalent to $\mathbf{S}^{|A|}$ and Lemma 2.6 implies that

$$
\Delta^{D'} \left( \mathbf{R}^{|A|}, \mathbf{S}^{|A|} \right) \leq \Gamma^{D'} \left( \mathbf{R}^{|A|} \right) = \Gamma \left[ D' \mathbf{S}^{|A|} \right] \left( \mathbf{R}^{|A|} \right) .
$$

Let us first consider the middle term $\Gamma^{D'} \left( \mathbf{R}^{|A|} \right)$. Note that the distinguisher $D'$ wins the game $\mathbf{R}^{|A|}$, where the restriction is seen as part of the output of $\mathbf{R}$, if and only if the original distinguisher $D$ wins the restricted game $\mathbf{R}^{|A|}$, i.e., $\Gamma^{D'} \left( \mathbf{R}^{|A|} \right) = \Gamma \left[ D \mathbf{S}^{|A|} \right] \left( \mathbf{R}^{|A|} \right)$.

Let us consider the last term $\Gamma \left[ D' \mathbf{S}^{|A|} \right] \left( \mathbf{R}^{|A|} \right)$. Note that the distinguisher $D'$ stops interacting with the system $\mathbf{S}^{|A|}$ once its restriction has been violated. In the random experiment formed by the distinguisher $\left[ D' \mathbf{S}^{|A|} \right]$ interacting with the game $\mathbf{R}^{|A|}$, both systems $\mathbf{R}^{|A|}$ and $\mathbf{S}^{|A|}$ receive the same inputs, and consequently the value of the restriction, which is a deterministic function of the inputs, is the same in both systems. Thus, the distinguisher $\left[ D' \mathbf{S}^{|A|} \right]$ stops interacting with the game $\mathbf{R}^{|A|}$ as soon as the restriction of the latter has been violated, and since the distinguisher
\([D'S^A] \) does not see the output of the game \(\overline{R^A} \) it follows that
\[
\Gamma[D'S^A](\overline{R^A}) = \hat{\Gamma}[D'S^A](\overline{R^A}).
\]

Finally, the distinguisher \(D' \) modifies the outputs of the system \(S^A \) to exactly correspond to what the system \( [S^A]^+ \) would have output when interacting with \( D \). Since all the queries issued by the distinguisher \([D'S^A] \) originate from the original distinguisher \( D \), it follows that
\[
\hat{\Gamma}[D'S^A](\overline{R^A}) = \hat{\Gamma}[D[S^A]^+](\overline{R^A}).
\]

The previous equations and the observation that \([D[S^A]^+] \) is a non-adaptive distinguisher lead to the desired result. \( \square \)

The previous theorem implies in particular that if a game is conditionally equivalent to some system, then the optimal strategy to win this game is non-adaptive.

**Corollary 4.1.** For any \((I \times X, Y)\)-game \(\overline{R} \),
\[
\text{CE} \left( \overline{R} \right) \implies \text{NA} \left( \overline{R} \right).
\]

**Proof.** Consider a number \( q \) of queries. Since the set \( \text{NA} \) of non-adaptive distinguishers is a subset of the set of all distinguishers, we trivially have \( \Gamma \left( \overline{R}^q \right) \geq \Gamma^{\text{NA}} \left( \overline{R}^q \right) \). If the predicate \( \text{CE} \left( \overline{R} \right) \) is true, then there exists some \((I \times X, Y)\)-system \( S \) such that \( \overline{R} \equiv S \). The restriction \( q \) on the total number of queries is a deterministic function of the inputs and Theorem 4.1 therefore implies \( \Gamma \left( \overline{R}^q \right) \leq \Gamma^{\text{NA}} \left( \overline{R}^q \right) \). Thus, \( \Gamma \left( \overline{R}^q \right) = \Gamma^{\text{NA}} \left( \overline{R}^q \right) \) for all \( q \in \mathbb{N} \) and \( \text{NA} \left( \overline{R} \right) \) follows. \( \square \)

### 4.2.2 CE and Disjunction of Games

We show in the next simple lemma that if one considers several games \(\overline{R}_1, \ldots, \overline{R}_r \), where each \(\overline{R}_j \) is an \((I \times X_j, Y_j)\)-game which is conditionally equivalent to some \((I \times X_j, Y_j)\)-system \( S_j \), then the disjunction \([\overline{R}_1, \ldots, \overline{R}_r] \) of these \( r \) games is conditionally equivalent to the combined system \([S_1, \ldots, S_r] \) formed by the parallel composition of these \( r \) systems.
Lemma 4.1. Consider \( r \) games \( \mathbf{R}_1, \ldots, \mathbf{R}_r \) and \( r \) systems \( \mathbf{S}_1, \ldots, \mathbf{S}_r \), where each \( \mathbf{R}_j \) is an \((I \times X_j, Y_j)\)-game and \( \mathbf{S}_j \) is an \((I \times X_j, Y_j)\)-system. Then,

\[
\mathbf{R}_1 \equiv \mathbf{S}_1 \land \cdots \land \mathbf{R}_r \equiv \mathbf{S}_r \implies \mathbf{[R_1, \ldots, R_r]} \equiv [\mathbf{S}_1, \ldots, \mathbf{S}_r].
\]

Proof. We shall denote by \( \mathbf{R} \) the game \( \mathbf{[R_1, \ldots, R_r]} \) and by \( \mathbf{S} \) the system \( [\mathbf{S}_1, \ldots, \mathbf{S}_r] \) within the remainder of the proof. The MBO of the game \( \mathbf{R} \) will be denoted by \( B \). Recall from Section 2.4.4 that \( \mathbf{R} \) is an \((I \times X, Y)\)-game and \( \mathbf{S} \) is an \((I \times X, Y)\)-system, where \( X := \bigsqcup_{j=1}^{r} X_j = \bigcup_{j=1}^{r} \{j\} \times X_j \) and \( Y := \bigsqcup_{j=1}^{r} Y_j = \bigcup_{j=1}^{r} \{j\} \times Y_j \). Consider a transcript \( T \) formed by \( k \) inputs \( (i_1, (j_1, x_1)), \ldots, (i_k, (j_k, x_k)) \in I \times X \) and \( k \) outputs \( (j_1, y_1), \ldots, (j_k, y_k) \) such that \( p_{Y_k | I_k X_k B_k = 0} (T) > 0 \) and therefore \( p_{Y_k | I_k X_k B_k = 0} \mathbf{R} \mathbf{|} (T) \) is defined. Such a transcript \( T \) can be partitioned into \( r \) sub-transcripts \( T_1, \ldots, T_r \) corresponding to response and query pairs made to the games \( \mathbf{R}_1, \ldots, \mathbf{R}_r \), respectively. Each sub-transcript \( T_j \) contains \( k_j \) queries, where \( k_1 + \cdots + k_r = k \), and we denote by \( B_j \) the MBO of the game \( \mathbf{R}_j \) in \( \mathbf{R} \). Note that \( p_{Y_k | I_k X_k B_k = 0} (T) \) being defined implies that \( p_{Y_k | I_k X_k B_k = 0} \mathbf{R} \mathbf{|} (T_j) \) is defined as well. Then,

\[
p_{Y_k | I_k X_k B_k = 0} \mathbf{R} (T) = \prod_{j=1}^{r} p_{Y_k | I_k X_k B_k = 0} \mathbf{R}_j (T_j)
\]

\[
= \prod_{j=1}^{r} p_{Y_k | I_k X_k B_k = 0} \mathbf{S}_j (T_j)
\]

\[
= p_{Y_k | I_k X_k} \mathbf{S} (T),
\]

where the first and last equality follow from the definition of parallel composition in Section 2.4.4, while the second equality follows from the fact that \( \mathbf{R}_j \equiv \mathbf{S}_j \), for all \( j \in \{1, \ldots, r\} \).

\(\square\)

4.2.3 NA and Disjunction of Games

Lemma 4.1 implies that if two games \( \mathbf{R}_1 \) and \( \mathbf{R}_2 \) are each conditionally equivalent to some system, then the combined game \( \mathbf{[R_1, R_2]} \) is also conditionally equivalent to some system. As we will show below, this property makes conditional equivalence a strictly stronger requirement than the optimal strategy for winning a game being non-adaptive.
Lemma 4.2. If the best strategies for winning the \((I \times X_1, Y_1)\)-game \(R_1\) and the \((I \times X_2, Y_2)\)-game \(R_2\) are both non-adaptive, then the best strategy for winning the game \([R_1, R_2]\) is not necessarily non-adaptive, i.e.,

\[
\text{NA}(R_1) \text{ and } \text{NA}(R_2) \implies \text{NA}([R_1, R_2]).
\]

Proof. We will define a sequence of games \(\{R_k\}_{k \geq 2}\) such that the implication \(\text{NA}(R_k) \implies \text{NA}([R_k, R_k])\) does not hold for any integer \(k \geq 2\). As a side result, we will have proved the stronger statement that adaptivity can help in any round when trying to win the game \([R_k, R_k]\).

Let \(k\) be an integer such that \(k \geq 2\) and let \(R_k\) be the following \((I \times \{0, 1\}, \{0, 1\})\)-game. Any query \(x_i \in \{0, 1\}\) made to \(R_k\) at any interface is responded at the same interface by a bit \(y_i \in \{0, 1\}\) chosen independently and uniformly at random. The monotone binary output of \(R_k\) is defined as \(A_1 := 0, \ldots, A_{k-1} := 0\), and for all \(l \geq k\), we have \(A_l := Y_1\), where \(Y_1\) is the response of \(R_k\) to the first query. We now show that the game \(R_k\) is such that

1. \(\hat{\Gamma}(R_k^{[q]}) = \hat{\Gamma}^\text{NA}(R_k^{[q]})\), for all \(q \in \mathbb{N}\);
2. \(\hat{\Gamma}([R_k, R_k]^{[k+1]}) > \hat{\Gamma}^\text{NA}([R_k, R_k]^{[k+1]})\).

Condition 1). In order to win the game \(R_k\), a game-winner simply needs to make at least \(k\) queries resulting in the event \(Y_1 = 1\). Since \(Y_1\), the answer of \(R_k\) to the first query, is an independent uniform random variable, the best strategy for winning the game \(R_k\) is clearly non-adaptive. Thus,

\[
\hat{\Gamma}(R_k^{[q]}) = \hat{\Gamma}^\text{NA}(R_k^{[q]}) = \begin{cases} 
0 & \text{if } q < k, \\
\frac{1}{2} & \text{otherwise}.
\end{cases}
\tag{4.1}
\]

Condition 2). Consider now a non-adaptive game-winner \(D\) trying to win the restricted game \([R_k, R_k]^{[k+1]}\) with at most \(k + 1\) queries. Since the game-winner \(D\) is non-adaptive, it has to fix all its queries in advance and in particular to which sub-game in \([R_k, R_k]^{[k+1]}\) each query is addressed to. Note that (4.1) implies that the game \(R_k\) cannot be won by any game-winner making strictly less than \(k\) queries. Thus,

\[
\hat{\Gamma}^\text{NA}([R_k, R_k]^{[k+1]}) = \frac{1}{2}.
\]
4.2 CE Versus NA

On the other hand, consider the following adaptive game-winner \(D\) making \(k+1\) queries to the restricted game \([R_k, R_k]^{[k+1]}\). The game-winner \(D\) makes a first query \((i, (1, x))\) to the game \([R_k, R_k]^{[k+1]}\), corresponding to a query \(x\) made to the interface \(i\) of the first sub-game \(R_k\) in \([R_k, R_k]^{[k+1]}\), which is replied by \((1, Y_1)\), for some \(i \in I, x \in \{0, 1\}\) and \(Y_1 \in \{0, 1\}\). If \(Y_1 = 1\), then \(D\) makes its remaining queries to the same sub-game it queried the first time by querying \(k\) times the game \([R_k, R_k]^{[k+1]}\) on \((i, (1, x))\). Otherwise, if \(Y_1 = 0\), then \(D\) makes its remaining \(k\) queries to the other sub-game by querying \(k\) times the game \([R_k, R_k]^{[k+1]}\) on \((i, (2, x))\). Note that in the case where \(Y_1 = 1\), which happens with probability \(\frac{1}{2}\), \(D\) wins the game \([R_k, R_k]^{[k+1]}\) with probability 1, while when \(Y_1 = 0\) the game-winner \(D\) wins the game \([R_k, R_k]^{[k+1]}\) with probability \(\frac{1}{2}\), i.e.,

\[
\hat{\Gamma}^D([R_k, R_k]^{[k+1]}) = \frac{3}{4} > \hat{\Gamma}^{\text{NA}}([R_k, R_k]^{[k+1]})
\]

The previous equation together with (4.1) imply that \(\text{NA}([R_k]) = 1\) and \(\text{NA}([R_k, R_k]) = 0\), for all \(k \geq 2\).

4.2.4 CE is Stronger

It is not hard to see from the previous results that conditional equivalence is strictly stronger a requirement than non-adaptive strategies being optimal for winning a game, a statement formalized below in Theorem 4.2. Assume for the sake of contradiction that for any game \(\overline{R}\),

\[
\text{NA}(\overline{R}) \implies \text{CE}(\overline{R}). \tag{4.2}
\]

Consider two games \(\overline{R}_1\) and \(\overline{R}_2\) such that \(\text{NA}(\overline{R}_1)\) and \(\text{NA}(\overline{R}_2)\), but not \(\text{NA}([\overline{R}_1, \overline{R}_2])\). Such games \(\overline{R}_1\) and \(\overline{R}_2\) must exist according to Lemma 4.2. Then, (4.2) implies \(\text{CE}(\overline{R}_1)\) and \(\text{CE}(\overline{R}_2)\), which in turn implies by Lemma 4.1 \(\text{CE}([\overline{R}_1, \overline{R}_2])\). The contradiction then follows from Corollary 4.1 which implies \(\text{NA}([\overline{R}_1, \overline{R}_2])\).

**Theorem 4.2.** There exists an \((I \times X, Y)\)-game \(\overline{R}\) such that the best strategy in winning \(\overline{R}\) is non-adaptive but there exists no \((I \times X, Y)\)-system \(S\) such that \(\overline{R}\) and \(S\) are conditionally equivalent, i.e.,

\[
\text{NA}(\overline{R}) \not\implies \text{CE}(\overline{R}).
\]
4.3 Parameterized Constructions from CE

In this section, we use the tools developed in Section 4.2 to derive uniform families of constructions parameterized by the number of queries that can be made to the assumed resource. To do so, it will be convenient to introduce the notion of query complexity to keep track of the number of queries made by a converter.

**Definition 4.2.** A converter $\alpha$ is said to have *query complexity* $\varphi$, where $\varphi : \mathbb{N} \rightarrow \mathbb{N}$, if for every $R \in \mathbb{N}$ and every $R$-tuple of outer queries, $\alpha$ does in total at most $\varphi(R)$ inner queries. We denote the query complexity of $\alpha$ as $\text{QC}_\alpha$.

It is easy to see that inner restrictions, i.e., restrictions on the number of queries that a converter can make to the attached resource, can be ignored if they are large enough, where large here is quantified by the query-complexity of the converter at hand. That is, for any number of queries $L$ and $R$, any converters $\alpha$ and $\beta$, and any system $R$, we have

$$\left[ L|\alpha R \right]^{-1} = \left[ L|\ell R \right]^{-1} \quad \text{and} \quad \left[ R|\beta^{\ell R} \right]^{-1} = \left[ R|\beta^{\ell R} \right]^{-1}, \quad (4.3)$$

for all $\ell \geq \text{QC}_\alpha(L)$ and $\rho \geq \text{QC}_\beta(R)$.

**4.3.1 Single Conditional Equivalence Statement**

Consider a protocol $\pi$ attached to the left-interface of a system $R$, resulting in the system $\pi R$, as well as a simulator $\sigma$ attached to the right-interface of a system $S$, resulting in the system $S \sigma$. Note that if an event is defined on the system $\pi R$, such that the resulting game $\pi R$ is conditionally equivalent to the system $S \sigma$, then Lemma 2.6 implies that the protocol $\pi$ constructs the resource $S$ from the assumed resource $R$ relative to the simulator $\sigma$. Alternatively, the same construction can be obtained if the event is instead defined on the system $S \sigma$ such that the resulting game $S \sigma$ is conditionally equivalent to the system $\pi R$. We show in Lemma 4.3 below that the same statement, i.e., conditional equivalence between a game $\pi R$ and a system $S \sigma$, also implies that the protocol $\pi$ constructs a family of restricted resources based on $S$ from a family of restricted resources based on $R$ and with respect to the simulator $\sigma$.

---

1As discussed in Section 2.5.2 this already assumes that the availability condition is satisfied, i.e., the converters and resources considered in Chapters 4 and 5 are such that $\pi R \perp \equiv S \perp$. 
Lemma 4.3. Consider a protocol $\pi$ and a simulator $\sigma$, as well as two systems $R$ and $S$. Let $\pi R$ denote the game formed by the system $\pi R$ adjoined with an arbitrary MBO. Then, the uniform family of protocols $\pi := \{L_\pi\}_{L,R \in \mathbb{N}}$ and the uniform family of simulators $\sigma := \{\sigma|_R\}_{L,R \in \mathbb{N}}$ are such that
\[\pi R \equiv S \sigma \quad \Rightarrow \quad \{\text{QC}_\pi(L)|_R\}_L \in \mathbb{N} \equiv \{\text{QC}_\pi(L)|_R\}_L,\]
where $\varepsilon_{L,R}(D) := \hat{\Delta}D'\left(\frac{L_\pi R}{R}\right)$ and $D'$ is a non-adaptive distinguisher such that $D' := \left[D\left[\frac{L|S}{L|\sigma}\right]^{-1}\right]$, for all $L, R \in \mathbb{N}$ and all distinguishers $D$.

Before proving Lemma 4.3, let us simply mention that the same result also holds if the assumed and desired worlds swap roles in the conditional equivalence statement, i.e., if the statement $S \sigma \equiv \pi R$ is assumed instead of $\pi R \equiv S \sigma$. Such a change of assumption only results in the appropriate change in the definition of $\varepsilon$ as follows, $\varepsilon_{L,R}(D) := \hat{\Delta}D'\left(\frac{L|S}{L|\sigma}\right)$ with $D' := \left[D\left[\frac{L|S}{L|\sigma}\right]^{-1}\right]$, for all $L, R \in \mathbb{N}$ and all distinguishers $D$.

Proof of Lemma 4.3. For every distinguisher $D$, we need to upper bound the restricted-distinguishing advantage between the query-restricted systems $L_\pi \text{QC}_{\pi}(L)|_R$ and $L_S \text{QC}_{\sigma}(R)|_R$, for all $L, R \in \mathbb{N}$. Notice that the inner restrictions, $\text{QC}_{\pi}(L)$ on the system $R$ and $\text{QC}_{\sigma}(R)$ on the system $S$, can be ignored without altering the restricted distinguishing advantage since those restrictions allow more queries than the converters will ever do, i.e., by (4.3) we have
\[\hat{\Delta}D'\left(\frac{L_\pi R}{R}, L_S \frac{\text{QC}_{\sigma}(R)}{\sigma}\right) \leq \hat{\Delta}D'\left(\frac{L_\pi R}{R}, L_S \frac{\sigma}{\sigma}\right),\]
for all distinguishers $D$.

Note that the restriction in both systems $L_\pi R|_R$ and $L_S \sigma|_R$ is a deterministic function of the system’s inputs, and by assumption the game $\pi R$ is conditionally equivalent to the system $S \sigma$. Therefore, Theorem 4.1 directly implies
\[\hat{\Delta}D'\left(\frac{L_\pi R}{R}, L_S \sigma|_R\right) \leq \hat{\Delta}D'\left(\frac{L_\pi R}{R}\right),\]
where $D' := \left[D\left[\frac{L|S}{L|\sigma}\right]^{-1}\right]$, for all $L, R \in \mathbb{N}$ and all distinguishers $D$.

Combining the two previous equations finishes the proof. \qed
4.3.2 Multiple Conditional Equivalence Statements

We show in Lemma 4.4 below that having multiple conditional equivalence statements allows to derive a family of parameterized constructions between resources which are globally restricted.

Constructions between globally-restricted resources. Each conditional equivalence statement taken separately implies via Lemma 4.3 a family of parameterized constructions and since the construction notion is composable in parallel (in the sense of Theorem 2.1) it follows that these multiple conditional equivalence statements do imply a family of parameterized constructions between resources taken in parallel, but which are however individually restricted. To illustrate this important point, let us first consider a family of parameterized constructions of the form

\[
\left\{ \left[ R_{L}^{L_{1}}, R_{R}^{R_{1}} \right] \right\}_{L_{1}, R_{1} \in \mathbb{N}} \xrightarrow{\text{IND}} \left\{ \left[ S_{L}^{L_{1}}, S_{R}^{R_{1}} \right] \right\}_{L_{1}, R_{1} \in \mathbb{N}},
\]

where the translation in the number of queries is given by \((\ell_{i}, r_{i}) := \varphi(L, R)\) for some function \(\varphi : \mathbb{N}^{2} \rightarrow \mathbb{N}^{2}\) and we assume for simplicity that the family \(\varepsilon = \{\varepsilon_{L,R}\}_{L,R \in \mathbb{N}}\) is such that \(\varepsilon_{L,R}(D) \leq \delta_{L,R}\) for some constant \(\delta_{L,R}\), for all \(L, R \in \mathbb{N}\) and all distinguishers \(D\). Consider now two instances of the same resource \(R\), both independent excepted for the number of queries that can be made to them. If we were to use the same protocol \(\pi\) on each instance, would we achieve the following construction

\[
\left\{ \left[ \left[ R_{L}^{L_{1}}, R_{R}^{R_{1}} \right] \right] \right\}_{L_{1}, R_{1} \in \mathbb{N}} \xrightarrow{\text{IND}} \left\{ \left[ \left[ S_{L}^{L_{1}}, S_{R}^{R_{1}} \right] \right] \right\}_{L_{1}, R_{1} \in \mathbb{N}},
\]

where the number of queries in the constructed resources evolve as described by (4.4)?

First, note that such a question, whether (4.4) implies (4.5), cannot be directly answered by standard parallel composition tools. Indeed, applying to (4.4) the fact that our security notion is composable in parallel would simply lead to a construction having twice as many parameters, where each resource would be individually restricted as follows

\[
\left\{ \left[ L_{1}^{L_{1}} R_{1}^{R_{1}}, L_{2}^{L_{2}} R_{2}^{R_{2}} \right] \right\}_{L_{1}, R_{1}, L_{2}, R_{2} \in \mathbb{N}} \xrightarrow{\text{IND}} \left\{ \left[ L_{1}^{L_{1}} S_{1}^{S_{1}}, L_{2}^{L_{2}} S_{2}^{S_{2}} \right] \right\}_{L_{1}, R_{1}, L_{2}, R_{2} \in \mathbb{N}}.
\]

Second, one would hope for a tight security bound in (4.5). That is, if a distinguisher were only able to make \(L_{1}\) (respectively, \(L_{2}\)) and \(R_{1}\) (respectively, \(R_{2}\)) queries to the left- and right-interface of the first
(respectively, second) resource in \( L|[\mathbf{R}, \mathbf{R}]|^{R} \), where \( L_1 + L_2 \leq L \) and \( R_1 + R_2 \leq R \), such a distinguisher would have an advantage in telling apart the real and ideal worlds in the construction stated in (4.5) of at most \( \delta_{L_1, R_1} + \delta_{L_2, R_2} \). In general, one would therefore hope that the distinguishing advantage \( \varepsilon'_{L, R}(D) \) is upper bounded for every distinguisher \( D \) by

\[
\max_{L_1 \in \{0, 1, \ldots, L\}, R_1 \in \{0, 1, \ldots, R\}} \delta_{L_1, R_1} + \delta_{L, L_1, R - R_1}.
\]

Unfortunately, such a tight bound is in general not true, i.e., there are cases, such as in the example given in the proof of Lemma 4.2, where choosing adaptively the number of queries does help in distinguishing. Note that Lemma 4.2 shows that even if the optimal strategy to distinguish \( \mathbf{S} \sigma \) from \( \mathbf{R} \) was non-adaptive, choosing adaptively the number of queries might still help in distinguishing \( L|[\mathbf{S}, \mathbf{S}]|^{\pi} \langle \sigma, \sigma \rangle \) from \( \langle \pi, \pi \rangle L|[\mathbf{R}, \mathbf{R}]|^{R} \).

In contrast, we show in the lemmas below that conditional equivalence is a sufficient criterion to obtain the desired family of constructions between \( r \) resources in parallel parameterized only by a pair of integers, and not by a \( 2r \)-tuple of integers as standard parallel composition would normally give us. Similarly to Lemma 4.3, the same construction can be obtained if the \( r \) conditional equivalence statements are in the reversed direction (the statement \( \overline{S}_j \sigma_j \equiv \pi_j \mathbf{R}_j \) is assumed instead of \( \overline{\pi_j \mathbf{R}_j} \equiv S_j \sigma_j \), for all \( j \in \{1, \ldots, r\} \)).

**Lemma 4.4.** Consider \( 2r \) systems \( \mathbf{R}_1, \ldots, \mathbf{R}_r \) and \( \mathbf{S}_1, \ldots, \mathbf{S}_r \), together with \( r \) protocols \( \pi_1, \ldots, \pi_r \) and \( r \) simulators \( \sigma_1, \ldots, \sigma_r \). Let \( \overline{\pi_j \mathbf{R}_j} \) denote the game formed by the system \( \pi_j \mathbf{R}_j \) adjoined with an arbitrary MBO such that

\[
\overline{\pi_j \mathbf{R}_j} \equiv S_j \sigma_j,
\]

for all \( j \in \{1, \ldots, r\} \).

Then, the uniform family of protocols \( \pi := \{ L|\pi \}_{L, R \in \mathbb{N}} \) and the uniform family of simulators \( \sigma := \{ \sigma R \}_{L, R \in \mathbb{N}} \), where \( \pi := \langle \pi_1, \ldots, \pi_r \rangle \) and \( \sigma := \langle \sigma_1, \ldots, \sigma_r \rangle \), are such that

\[
\left\{ \text{QC}_{\pi}(L)|[\mathbf{R}_1, \ldots, \mathbf{R}_r]|^{R} \right\}_{L, R \in \mathbb{N}} \overset{(\pi, \sigma, \varepsilon)\text{ IND}}{\mapsto} \left\{ L|[\mathbf{S}_1, \ldots, \mathbf{S}_r]|^{\text{QC}_{\sigma}(R)} \right\}_{L, R \in \mathbb{N}},
\]

where \( \varepsilon_{L, R}(D) := \hat{\Gamma}^{D'} \left( L|[\pi_1 \mathbf{R}_1, \ldots, \pi_r \mathbf{R}_r]|^{R} \right) \) and \( D' \) is a non-adaptive distinguisher such that \( D' := \left[ D \left[ L|[\mathbf{S}_1, \ldots, \mathbf{S}_r]| \sigma R \right]^{-1} \right] \), for all \( L, R \in \mathbb{N} \) and all distinguishers \( D \).
Proof. For each \( j \in \{1, \ldots, r\} \), the game \( \pi_j R_j \) is assumed to be conditionally equivalent to the system \( S_j \sigma_j \). Therefore, Lemma 4.1 implies that the disjunction of these \( r \) games \( [\pi_1 R_1, \ldots, \pi_r R_r] \) is conditionally equivalent to the combined system \([S_1 \sigma_1, \ldots, S_r \sigma_r]\). The definition of parallel composition and (2.2) in Section 2.4.4 imply that the combined system \([S_1 \sigma_1, \ldots, S_r \sigma_r]\) can be simply rewritten as \( S \sigma \), where \( S := [S_1, \ldots, S_r] \).

Similarly, the system \([\pi_1 R_1, \ldots, \pi_r R_r]\) can be rewritten as \( \pi R \), where \( R := [R_1, \ldots, R_r] \). We can therefore add an MBO to the system \( \pi R \), obtaining the game \( \pi' R' \), such that the game is won if and only one of the games \( \pi_j R_j \) is won, i.e.,

\[
\pi' R' = [\pi_1 R_1, \ldots, \pi_r R_r].
\]

Overall, the game \( \pi' R' \) is thus conditionally equivalent to the system \( S \sigma \) and Lemma 4.3 implies the corresponding family of parameterized constructions

\[
\left\{ QC_{\pi}(L) R^{|R|} \right\}_{L,R \in \mathbb{N}} \xrightarrow{\text{IND}} \left\{ L|S QC_{\sigma}(R) \right\}_{L,R \in \mathbb{N}},
\]

where \( \varepsilon_{L,R}(D) := \hat{\Gamma}^D \left( L|\pi' R'| \right) \), for all \( L, R \in \mathbb{N} \) and all distinguishers \( D \).

\( \square \)

Measures on systems composed in parallel. We show that quantities related to systems composed in parallel, such as the query complexity of a converter formed by the parallel composition of others or the probability of winning the disjunction of multiple games in a non-adaptive manner appearing in Lemma 4.4, can be upper bounded by that of the individual systems. In particular, the bound on the distinguishing advantage obtained in Lemma 4.4 depends on the the probability of winning the disjunction of \( r \) games in a non-adaptive manner. The lemma below shows that this corresponds to the desired tight distinguishing bound described in (4.6).

Lemma 4.5. Let \( R_j \) be an \((\mathcal{I} \times \mathcal{X}_j, \mathcal{Y}_j)\)-game, for all \( j \in \{1, \ldots, r\} \). Then, for every non-adaptive distinguisher \( D \in NA \)

\[
\hat{\Gamma}^D \left( L|R_1, \ldots, R_r |R \right) \leq \max \hat{\Gamma}^{D_1} \left( L_1|R_1, R_1 \right) + \cdots + \hat{\Gamma}^{D_r} \left( L_r|R_r, R_r \right),
\]

where \( D_1, \ldots, D_r \) are non-adaptive distinguishers described in the proof below and the maximum is over all integers \( L_1, \ldots, L_r \) and \( R_1, \ldots, R_r \) such that \( L_1 + \cdots + L_r \leq L \) and \( R_1 + \cdots + R_r \leq R \).
4.3 Parameterized Constructions from CE

Proof. In order to have shorter notations within the proof, let us denote by $R$ the system $[R_1, \ldots, R_r]$. Consider a non-adaptive game winner $D$ trying to win the restricted game $\overline{L|R}$. Conditioned on its internal randomness, the game winner $D$ does $L_j$ and $R_j$ queries to the left- and right- interface of the game $R_j$ in the disjunction $\overline{L|R}$, respectively, for some integers $L_1, \ldots, L_r$ and $R_1, \ldots, R_r$ such that $L_1 + \cdots + L_r \leq L$ and $R_1 + \cdots + R_r \leq R$. Thus, winning the disjunction $\overline{L|R}$ requires to win one of games $\overline{L_1|R_1} \text{ or } \overline{L_2|R_2}$ in $\overline{L|R}$, for some $j \in \{1, \ldots, r\}$. Let $D_j$ be the distinguisher obtained from $D$ by forwarding any query and response made to the left$_j$- or right$_j$-interface, while other queries are replied as in $\overline{L|R}$, i.e., $D_j := D^{L_j|R_j}$. Then, the union bound implies that

$$ \hat{\Gamma}^D(\overline{L|R}) \leq \max \hat{\Gamma}^{D_1}(\overline{L_1|R_1}) + \cdots + \hat{\Gamma}^{D_r}(\overline{L_r|R_r}) $$

where the maximum is over all integers $L_1, \ldots, L_r$ and $R_1, \ldots, R_r$ such that $L_1 + \cdots + L_r \leq L$ and $R_1 + \cdots + R_r \leq R$. □

The query complexity of the protocol $\langle \pi_1, \ldots, \pi_r \rangle$ and of the simulator $\langle \sigma_1, \ldots, \sigma_r \rangle$ appearing in the construction stated in Lemma 4.4 can easily be upper bounded by a function of the query complexity of each individual converter by using the next simple lemma.

Lemma 4.6. For any $r$ converters $\alpha_1, \ldots, \alpha_r$

$$ \text{QC}_{\langle \alpha_1, \ldots, \alpha_r \rangle}(k) \leq \max \text{QC}_{\alpha_1}(k_1) + \cdots + \text{QC}_{\alpha_r}(k_r) $$

where the maximum is taken over all integers $k_1, \ldots, k_r$ summing up to $k$, for all integers $k$.

Proof. Consider $k$ queries made to the outside interface of the combined converter $\langle \alpha_1, \ldots, \alpha_r \rangle$, for some integer $k$. Those $k$ queries can be partitioned into $k_1, \ldots, k_r$ queries made to the outside interface of each converter $\alpha_1, \ldots, \alpha_r$, respectively, where $k_1, \ldots, k_r$ are non-negative integers such that $k_1 + \cdots + k_r = k$. Each converter $\alpha_j$ does at most $\text{QC}_{\alpha_j}(k_j)$ inner queries, for all $j \in \{1, \ldots, r\}$, and the combined converter $\langle \alpha_1, \ldots, \alpha_r \rangle$ does therefore at most $\text{QC}_{\alpha_1}(k_1) + \cdots + \text{QC}_{\alpha_r}(k_r)$. □
Chapter 5

Query-Complexity Amplification

In this chapter, we consider the task of amplifying the query-complexity of a random oracle that we formally define in Section 5.1. We then show in Section 5.2 that the natural protocol consisting of iterating a constant number of times a random oracle fails to be a query-complexity amplifier. Consequently, we provide in Section 5.3.1 a simple scheme, called collision-free iteration, that is proven in Section 5.3.2 to achieve the desired notion of query-complexity amplification. Finally, we show in Section 5.3.3 that multiple independent instances of the same collision-free iteration protocol can be used to globally amplify the query-complexity of multiple random oracles.

This chapter is stated in the indifferentiability setting described in Section 2.5.2 and uses the formalism of query-restricted systems developed in Section 4.2.1. In particular, resources in this chapter have two interfaces, a left- and a right-interface, and the set \( I \) of interfaces is \( I := \{ \text{left}, \text{right} \} \).

5.1 QCA for Random Oracles

5.1.1 Query-Complexity Amplifiers

As outlined in Section 1.2.2, we formalize query-complexity amplification (QCA) as a construction of random oracles which only allow for a limited number of queries from random oracles which allow more queries, both at
the (honest user’s) left and at the (adversary’s) right interface. That is, we consider a random oracle as a resource, and the “quality” of a certain QCA scheme will be captured by the translation of restrictions (in the numbers of queries) that it achieves at both the honest and the adversarial interface. In this section we formalize the above intuition. Recall from Section 2.6.1 that $RO$ denotes a random oracle with output length $n$.

**Definition 5.1.** Consider a function $\varphi : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$ and a family $\varepsilon := \{\varepsilon_{L,R}\}_{L,R \in \mathbb{N}}$ where each $\varepsilon_{L,R}$ is a function mapping each distinguisher to a real number in $[0, 1]$. Then, a uniform family of protocols $\{\pi_{L,R}\}_{L,R \in \mathbb{N}}$, where $\pi_{L,R} = \pi$ for all $L, R \in \mathbb{N}$ and for some *deterministic* and *stateless* protocol $\pi \in \Sigma$, is said to be a $(\varphi, \varepsilon)$-*query-complexity amplifier*, with respect to a family of simulators $\sigma := \{\sigma_{L,R}\}_{L,R \in \mathbb{N}}$, if

$$\left\{ L^{|RO^{|R}} \right\}_{L,R \in \mathbb{N}} \overset{(\pi, \sigma, \varepsilon)}{\xrightarrow{\text{IND}}} \left\{ L^{|RO^{|n}} \right\}_{L,R \in \mathbb{N}},$$

where $(\ell, n) := \varphi (L, R)$ and $n < R$, for all $L, R \in \mathbb{N}$.

According to Definition 5.1, proving that a protocol $\pi$ is a $(\varphi, \varepsilon)$-query-complexity amplifier requires in particular to show that the system $\pi_{L,R}^{|RO^{|R}}$ is within $\varepsilon_{L,R} (D)$ from the system $\ell^{|RO^{|n}} \sigma_{L,R}$, and where $(\ell, n) := \varphi (L, R)$ quantifies the exact amplification achieved for all $L, R \in \mathbb{N}$ and all distinguishers $D$. Both resources are depicted in Figure 5.1. In practice, where random oracles are instead actual hash functions, the goal of a query-complexity amplifier is therefore to transform a hash function into a new harder-to-evaluate hash function. Additionally requiring that a query-complexity amplifier be both deterministic and stateless\(^1\) ensures that the result of this transformation is indeed a function\(^2\).

### 5.1.2 Number of Queries

It may not be directly apparent from the definition of a query-complexity amplifier stated in Definition 5.1 that in order to evaluate such a scheme,

---

\(^1\)A converter is said to be stateless if it does not keep a state between answering outer queries, i.e., its behavior for a particular outer query depends only on the query itself and the ongoing interaction at the inside interface. We refer to [DGHM13, Def. 1] for a more formal treatment.

\(^2\)Definition 5.1 is a slight modification from the corresponding definition in the proceedings version of this work [DGHT15], where it did not explicitly require the converter to be deterministic and stateless. Without this requirement, the (probabilistic and stateful) converter that does lazy sampling, i.e., it samples a fresh random value for each new query, would perfectly achieve any query-complexity amplification.
5.1 QCA for Random Oracles

(a) Real world with the assumed resource $^L\text{RO}^R$.

(b) Ideal world with the desired resource $^L\text{RO}^R$.

**Figure 5.1.** A $(\varphi, \varepsilon)$-query-complexity amplifier $\pi$: For any number of queries $L, R$, the resource on the left is within $\varepsilon (L, R)$ from the resource on the right and the simulator $\sigma_{L,R}$ does at most $\mathcal{r} < R$ inner queries, where $(\ell, \mathcal{r}) := \varphi (L, R)$.

A certain minimum amount of queries to the underlying random oracle is *unavoidable*. To formalize this statement, we consider in this section a game $\overline{Q}$, which is parameterized by a query-complexity amplifier $\pi$ and two integers $\ell$ and $\mathcal{r}$. Intuitively, the game $\overline{Q} (\pi, \ell, \mathcal{r})$ is won if $\ell$ outputs of the query-complexity amplifier $\pi$ can be determined by doing at most $\mathcal{r}$ queries to the underlying random oracle. To do so, the game $\overline{Q} (\pi, \ell, \mathcal{r})$ emulates internally $[\pi \text{RO}^\mathcal{r}]^\dagger$, a random oracle restricted to at most $\mathcal{r}$ adversarial queries and with the query-complexity amplifier $\pi$ attached to its left-interface. Queries made to the right-interface of the game $\overline{Q} (\pi, \ell, \mathcal{r})$ are answered by the (internally emulated) random oracle, while at the left-interface the game $\overline{Q} (\pi, \ell, \mathcal{r})$ expects $\ell$ pairs $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ and $\overline{Q} (\pi, \ell, \mathcal{r})$ checks for each pair whether $y_j$ is indeed the output of the query-complexity amplifier $\pi$ on the specified input $x_j$. Formally, the game $\overline{Q} (\pi, \ell, \mathcal{r})$ corresponds to the system $Q (\pi, \ell, \mathcal{r})$ detailed in System 5.1 and where the MBO is indicated by the value of $\text{win}$.

**Constant-fold query-complexity amplifiers.** Before proceeding any further, let us consider for a moment the special case of $c$-fold query-complexity amplifiers, such as the collision-free iteration protocol $\text{amp}_c$ described in Section 5.3, for some integer $c$. The next lemma shows that in order to successfully compute $\ell$ evaluations of a $c$-fold query-complexity amplifier, at least $\ell c$ queries to the underlying random oracle are required.

**Lemma 5.1.** Let $\{\pi_{L,R}\}_{L,R \in \mathbb{N}}$ be a $(\varphi, \varepsilon)$-query-complexity amplifier, with respect to some family of simulators $\{\sigma_{L,R}\}_{L,R \in \mathbb{N}}$, where $\pi_{L,R}^{-} = \pi$ and $\varphi (L, R) := (\lfloor \frac{L}{c} \rfloor, \lfloor \frac{R}{c} \rfloor)$ for all $L, R \in \mathbb{N}$. Then, for any integer $\ell$ and for
System 5.1: Game \( Q(\pi, \ell, n) \) to count the number of queries

\[
\text{win} := 0
\]

on input \( x \in \{0, 1\}^* \) at right

\[
y := \text{emulate result of querying } x \text{ at the right-interface of } \left[ \pi \text{RO}\right]^r
\]

output \( y \) at right

on first input \((x_1, y_1), \ldots, (x_\ell, y_\ell) \in \{0, 1\}^* \times \{0, 1\}^{\ell_c} \) at left

if \( x_1, \ldots, x_\ell \) are distinct then

\[
\text{for } 1 \leq j \leq \ell \text{ do}
\]

\[
y'_j := \text{emulate result of querying } x_j \text{ at the left-interface of } \left[ \pi \text{RO}\right]^l
\]

\[
\text{win} := (y_1 = y'_1) \land \cdots \land (y_\ell = y'_\ell)
\]

output \( 0^n \) at left

any game winner \( D \), there exists a distinguisher \( D' \) such that

\[
\Gamma^D \left( \overline{Q}(\pi, \ell, \ell c - 1) \right) \leq \epsilon_{\ell c, \ell c - 1}(D') + 2^{-n},
\]

where the distinguisher \( D' \) does the same queries as \( D \) and is described in System 5.2.

Proof. Consider a number \( \ell \) of queries and a game winner \( D \) for the game \( \overline{Q}(\pi, \ell, \ell c - 1) \). Note that the game \( \overline{Q}(\pi, \ell, \ell c - 1) \) is irremeridibly lost if more that \( \ell c - 1 \) queries are made to its right-interface and we therefore assume without loss of generality that the game winner \( D \) does at most \( \ell c - 1 \) queries at the right-interface. We transform the game winner \( D \) into a distinguisher \( D' \) trying to tell apart the restricted systems \( \pi_{\ell c, \ell c - 1} \text{RO}^{\ell c - 1} \) and \( \ell \text{RO}^{\ell - 1} \sigma_{\ell c, \ell c - 1} \), that we will denote within the proof by \( R_\ell \) and \( S_\ell \), respectively. The distinguisher \( D' \) proceeds in two phases as follows:

1. Every right-interface query made by the game winner \( D \) is forwarded to the right-interface of the connected system (\( R_\ell \) or \( S_\ell \)). If during that phase a query is responded by the error symbol \( \bullet \), indicating that one of the MBOs went off, the distinguisher \( D' \) returns the value 0 as its decision bit and halts.

2. When the game winner \( D \) outputs \( \ell \) pairs \((x_1, y_1), \ldots, (x_\ell, y_\ell)\), the distinguisher \( D' \) queries the left-interface of the connected system.
on $x_j$ to obtain the response $y'_j$, for every $j \in \{1, \ldots, \ell\}$. The distinguisher $D'$ returns 1 as its decision bit if one of the following two events happens: $y'_j = \bullet$ for some $j \in \{1, \ldots, \ell\}$; or, all the bit strings $x_1, \ldots, x_\ell$ are distinct and $y'_j = y_j$ for all $j \in \{1, \ldots, \ell\}$.

Note that the distinguisher $D'$ does exactly the same queries as the original game winner $D$. The distinguisher $D'$ is detailed in System 5.2.

**System 5.2: Distinguisher $D'$ based on game winner $D$**

\[
\begin{align*}
  &x_j := \diamond, y_j := \diamond \text{ and } y'_j := \diamond \text{ for all } j \in \{1, \ldots, \ell\} \\
  & (i, t) := 1^{\text{st}} \text{ output of } D \\
  \text{while } i = \text{right} \text{ do} \\
  & y := \text{result of querying } t \text{ at right} \\
  & \quad \text{if } y = \bullet \text{ then return } 0 \\
  & \quad (i, t) := \text{emulate result of querying } D \text{ on } (\text{right}, y) \\
  \text{parse } t \text{ as } (x_1, y_1), \ldots, (x_\ell, y_\ell) \\
  \text{for } 1 \leq j \leq \ell \text{ do} \\
  & y'_j := \text{result of querying } x_j \text{ at left} \\
  & \quad \text{if } y'_j = \bullet \text{ then return } 1 \\
  \text{return } (|\{x_1, \ldots, x_\ell\}| = \ell) \land (y'_1 = y_1) \land \cdots \land (y'_\ell = y_\ell)
\end{align*}
\]

We now lower bound the distinguishing advantage $\Delta D' (R_\ell, S_\ell)$. First, we upper bound the probability $P^{D'S_\ell \leftarrow} (B = 1)$ of the distinguisher $D'$ outputting 1 in the $DS_\ell \leftarrow$ random experiment. Note that the distinguisher $D'$ can only output 1 when interacting with $S_\ell$ if the MBO of $S_\ell$ did not go off at the end of the first phase. Thus, all the responses $y'_1, \ldots, y'_\ell$ obtained at the end of the second phase are different from the error symbol $\bullet$ since the queries are directly made to the left-interface of the random oracle $\ell RO^{\ell-1} | \ell$ in $S_\ell$. In the $DS_\ell \leftarrow$ random experiment, the simulator can do at most $\ell - 1$ queries to its random oracle, and therefore when the distinguisher queries the left-interface of the random oracle on the distinct inputs $x_1, \ldots, x_\ell$ at least one of them was never queried before to the random oracle and thus $P^{DS_\ell \leftarrow} (B = 1) \leq 2^{-n}$.

Second, we lower bound the probability $P^{D'R_\ell \leftarrow} (B = 1)$ of the distinguisher $D'$ outputting 1 in the $DR_\ell \leftarrow$ random experiment. Note that queries made to the right-interface are answered directly by the random oracle $\ell RO^{\ell c-1} | \ell c$ in $R_\ell$ so that in the $DR_\ell \leftarrow$ random experiment the distinguisher $D'$ never outputs 0 at the end of the first phase. Therefore, if none of the outputs $y'_j$ obtained during the second phase are the error
symbol ♦, the distinguisher \(D'\) outputs 1 if the original game winner \(D\) would have won the game \(Q(\pi, \ell, \ell c - 1)\); otherwise, if \(y'_j = ♦\) for some \(j \in \{1, \ldots, \ell\}\) the distinguisher \(D'\) outputs 1 with probability \(1\). Thus, \(P_{D' \leftarrow \mathcal{R}^{\ell - 1}}(B = 1) \geq \Gamma^D (Q(\pi, \ell, \ell c - 1))\) and together with the previous upper bound on \(P_{D' \leftarrow \mathcal{S}^{\ell - 1}}(B = 1)\) we have

\[
\hat{\Delta}^{D'}(R, S) \geq \Gamma^D (Q(\pi, \ell, \ell c - 1)) - 2^{-n}.
\]

Finally, since the family \(\{\pi_{L,R}\}_{L,R \in \mathbb{N}}\) is a \(c\)-fold query-complexity amplifier by assumption, the query-restricted random oracle \(q^c \mathcal{R}^{\ell - 1}\) can therefore be constructed from \(q^c \mathcal{R}^{\ell - 1}\). That is, Definition 5.1 implies that for any distinguisher \(D'\), it holds

\[
\varepsilon_{\ell c, \ell c - 1}(D') \geq \hat{\Delta}^{D'}(R, S).
\]

Combining the two previous equations finishes the proof. \(\square\)

**More general query-complexity amplifiers.** The above argument, which solely dealt with \(c\)-fold query-complexity amplifiers, can be generalized to a broad class of \((\varphi, \varepsilon)\)-query-complexity amplifiers under the following mild assumptions on the function \(\varphi\):

1. \(\varphi\) is surjective, i.e., for every pair of integers \((\ell, \ell')\), there exists another pair of integers \((L, R)\) such that \(\varphi(L, R) = (\ell, \ell')\);

2. \(\varphi\) is non-decreasing with respect to both parameters, i.e., if \(L' \geq L\) (respectively, \(R' \geq R\)), then \(\ell' \geq \ell\) (respectively, \(\ell' \geq \ell\)), where \((\ell, \ell') := \varphi(L, R)\) and \((\ell', \ell') := \varphi(L', R')\), for all integers \(L, R, L', R'\).

Since the function \(\varphi\) is surjective, there exist pairs of integers \((L, R)\) mapping to \((m, \ell)\), for some fixed integer \(\ell\) and some \(m \geq \ell\). Let us denote by \(R_{\min}(\ell)\) the minimum number of adversarial queries \(R\) in such pairs, i.e.,

\[
R_{\min}(\ell) := \min \{ R \in \mathbb{N} \mid \exists m, L \in \mathbb{N} : m \geq \ell \text{ and } \varphi(L, R) = (m, \ell) \}.
\]

Similarly, it will be convenient to denote by \(L_{\min}(\ell)\) the minimum number of honest queries which map via the function \(\varphi\) to at least \(\ell\) honest queries and exactly \(\ell\) adversarial queries, when the initial number of adversarial queries is minimum, i.e.,

\[
L_{\min}(\ell) := \min \{ L \in \mathbb{N} \mid \exists m \in \mathbb{N} : m \geq \ell \text{ and } \varphi(L, R_{\min}(\ell)) = (m, \ell)\}.
\]
for all $\ell \in \mathbb{N}$. In the following lemma, we show that any $\ell$ evaluations of a query-complexity amplifier requires at least $R_{\min}(\ell)$ queries to the underlying random oracle, where of course $R_{\min}(\ell) > \ell$. Note that in Lemma 5.1, where we considered solely a $c$-fold query-complexity amplifier, the values $R_{\min}(\ell)$ and $L_{\min}(\ell)$ corresponded to $\ell c$.

**Lemma 5.2.** Let \( \{\pi_{L,R}\}_{L,R \in \mathbb{N}} \) be a \((\varphi, \varepsilon)\)-query-complexity amplifier, with respect to some family of simulators \( \{\sigma_{L,R}\}_{L,R \in \mathbb{N}} \), where \( \pi_{L,R}^- = \pi \) for all \( L, R \in \mathbb{N} \) and \( \varphi \) is surjective and non-decreasing with respect to both parameters as described above. Then, for any integer \( \ell \) and for any game winner \( \mathbf{D} \), there exists a distinguisher \( \mathbf{D}' \) such that

\[
\Gamma^D\left(\overline{Q}(\pi, \ell, R_{\min}(\ell) - 1)\right) \leq \varepsilon_{L_{\min}(\ell), R_{\min}(\ell) - 1}(\mathbf{D}') + 2^{-n},
\]

where the distinguisher \( \mathbf{D}' \) does the same queries as \( \mathbf{D} \) and is described in System 5.2.

**Proof.** The proof is almost identical to that of Lemma 5.1 and we therefore omit redundant arguments. Consider a number \( \ell \) of queries and a game winner \( \mathbf{D} \) for the game \( \overline{Q}(\pi, \ell, R_{\min}(\ell) - 1) \). Exactly as in the proof of Lemma 5.1, the game winner \( \mathbf{D} \) can be transformed in a distinguisher \( \mathbf{D}' \) given in System 5.2 trying to tell apart the restricted systems \( \pi_{L_{\min}(\ell), R_{\min}(\ell) - 1}^{L_{\min}(\ell)} \overline{Q}^{R_{\min}(\ell) - 1} \) and \( \ell' \overline{Q}^{\varphi_{L_{\min}(\ell), R_{\min}(\ell) - 1}} \), that will be denoted within the proof by \( \mathbf{R}_\ell \) and \( \mathbf{S}_\ell \), respectively, with \( (\ell', \ell') := \varphi(L_{\min}(\ell), R_{\min}(\ell) - 1) \).

Note that by definition of the integers \( R_{\min}(\ell) \) and \( L_{\min}(\ell) \), it follows that \( \varphi(L_{\min}(\ell), R_{\min}(\ell)) = (m, \ell) \), for some integer \( m \) such that \( m \geq \ell \). Furthermore, the parameters \( \ell' \) and \( \ell' \) have the following properties. First, note that \( \ell' \geq m \geq \ell \) and \( \ell' \leq \ell \) since the function \( \varphi \) is non-decreasing with respect to both parameters. Second, \( \ell' \neq \ell \) (and thus \( \ell' < \ell \leq \ell' \)) since \( R_{\min}(\ell) \) is the minimum number of adversarial queries among all pairs mapping via the function \( \varphi \) to at least \( \ell \) honest queries and exactly \( \ell \) adversarial queries. Then, the same arguments used in the proof of Lemma 5.1 show, very roughly, that the distinguisher \( \mathbf{D}' \) can determine \( \ell \) outputs of the query-complexity amplifier \( \pi \) when interacting with \( \mathbf{R}_\ell \), but not when interacting with \( \mathbf{S}_\ell \) since the simulator can only ask \( \ell' < \ell \) queries to its random oracle. Thus,

\[
\hat{\Delta}^{\mathbf{D}'}(\mathbf{R}_\ell, \mathbf{S}_\ell) \geq \Gamma^D\left(\overline{Q}(\pi, \ell, L_{\min}(\ell) - 1)\right) - 2^{-n}.
\]

Finally, since the family \( \{\pi_{L,R}\}_{L,R \in \mathbb{N}} \) is a \((\varphi, \varepsilon)\)-query-complexity amplifier it follows that for any distinguisher \( \mathbf{D}' \)

\[
\varepsilon_{L_{\min}(\ell), R_{\min}(\ell) - 1}(\mathbf{D}') \geq \hat{\Delta}^{\mathbf{D}'}(\mathbf{R}_\ell, \mathbf{S}_\ell).
\]
The proof is then finished by combining the two previous equations. □

5.2 The Caveats of Plain Iterated Hashing

We show in this section that the protocol consisting of iterating \( c \) times a random oracle, denoted \( \text{iter}_c \), is not a query-complexity amplifier, for any number \( c \geq 2 \) of iteration. To do so, we generalize some of the results of Dodis et al. [DRST12], who specifically focused on the case \( c = 2 \), to deal with a higher number of iterations.

5.2.1 Failure as a Query-Complexity Amplifier

The next theorem shows that if one assumes a random oracle with only 2 adversarial queries, then the random oracle constructed by the \( c \)-iteration protocol \( \text{iter}_c \) must allow at least \( \ell \) adversarial queries, where \( \ell \) roughly corresponds to the number of honest queries in the constructed random oracle. This implies therefore that the \( c \)-iteration protocol \( \text{iter}_c \) is not a query-complexity amplifier according to Definition 5.1 (unless the distinguishing advantage becomes trivial).

To give some intuition behind this result, consider the \( c \)-iteration of a random oracle \( \text{iter}_c \text{RO} \) and a chain \( (y^{(0)}, y^{(c)}, \ldots, y^{(c\ell)}) \) of \( \ell \) hashes, where \( y^{(cj)} \) denotes the output of the \( c \)-iteration protocol \( \text{iter}_c \) when queried on the previous chain element \( y^{(c(j-1))} \). The key observation here is that \( y^{(c\ell+1)} \), the output of the random oracle \( \text{RO} \) when queried on the last chain element \( y^{(c\ell)} \), forms the end of another chain of \( \ell \) hashes starting with \( y^{(1)} \), the output of \( \text{RO} \) when queried on the first element \( y^{(0)} \) of the previous chain, and that both chains do not have any element in common (with overwhelming probability). In contrast, such shifted chains of queries cannot occur in the system \( \text{RO} \sigma \), unless the simulator \( \sigma \) does at least \( \ell \) inner queries to its underlying random oracle.

Note that if the assumed random oracle in Theorem 5.1 had more adversarial queries, say \( R \) instead of 2, then one could force the simulator to make in total in the order of \( \Omega(\ell R) \) queries to the underlying random oracle by “hiding” the query on the last chain element \( y^{(c\ell)} \) among \( R - 2 \) random queries. A similar technique was used in [DRST12, Th. 1].

**Theorem 5.1.** The protocol \( \text{iter}_c \), consisting of iterating \( c \) times a random oracle, where \( c \geq 2 \), is such that for any number \( \ell \) of queries, there exists a distinguisher \( D_{\text{iter}_c,\ell,n} \) given in System 5.3 such that for any simulator \( \sigma \),

\[
2c\ell \text{RO}^2 \xrightarrow{\text{(iter}_c,\sigma,\epsilon)} 2\ell \text{RO}^n \quad \Rightarrow \quad \forall n \geq \ell \quad \exists \left( D_{\text{iter}_c,\ell,n} \right) \geq 1 - \mu,
\]
where \( \mu := 2^{-n} \cdot f(c, \ell) \) and \( f(c, \ell) := \frac{1}{1-3\ell} 2^{-n} + \frac{1}{2} (3\ell)^2 + 2 (c\ell + 1)^2 \).

**Proof.** Let us assume that \( n < \ell \) since otherwise the proof is finished. In order to have shorter notations within the proof, let us denote by \( \mathbf{R} \) and \( \mathbf{S} \) the systems \( \text{iter}_{c} \, 2^{c\ell} \mathbf{RO}^{2} \) and \( 2^{c\ell} \mathbf{RO}^{n} \, \sigma \), respectively, for some simulator \( \sigma \). Then, we give a distinguisher \( \mathbf{D}_{\ell, n}^{\text{iter}} \), described in System 5.3, and show that it achieves the desired restricted distinguishing advantage, i.e., \( \hat{\Delta} \mathbf{D}_{\ell, n}^{\text{iter}} (\mathbf{R}, \mathbf{S}) \geq 1 - \mu \). Intuitively, the distinguisher \( \mathbf{D}_{\ell, n}^{\text{iter}} \) is based on the aforementioned shifted chains of queries. In greater details, the distinguisher \( \mathbf{D}_{\ell, n}^{\text{iter}} \) first prepares an \( \ell \)-chain \( (y(0), y(c), \ldots, y(c\ell)) \) of hashes, starting at a random \((n + 1)\)-bit string \( y(0) \), by querying \( \ell \) times the left-interface of \( \mathbf{R} \) or \( \mathbf{S} \). The fact that \( y(0) \) is an \((n + 1)\)-bit string instead of an \( n \)-bit string as the rest of the chain elements \( y(c), \ldots, y(c\ell) \) will make the analysis easier. Note that when interacting with the system \( \mathbf{R} \), the chain element \( y(cj) \) is indeed the \( c \)-iterate of the random oracle when queried on the previous chain element \( y(c(j-1)) \), for all \( j \in \{1, \ldots, \ell \} \). Then, the distinguisher \( \mathbf{D}_{\ell, n}^{\text{iter}} \) tries to “shift” the obtained chain by querying successively \( y(c\ell) \) and \( y(0) \) at the right-interface to obtain \( y(c\ell + 1) \) and \( y(1) \), respectively. When interacting with \( \mathbf{R} \), the values \( y(c\ell + 1) \) and \( y(1) \) are simply the answers of the random oracle when queried on \( y(c\ell) \) and \( y(0) \), respectively. Finally, the distinguisher \( \mathbf{D}_{\ell, n}^{\text{iter}} \) checks by doing \( \ell \) queries at the left-interface that \( y(c\ell + 1) \) is indeed the end of an \( \ell \)-chain of hashes starting with \( y(1) \) and that the shifted chain does not have any element in common with the chain initially prepared. Overall, the distinguisher \( \mathbf{D}_{\ell, n}^{\text{iter}} \) does \( 2\ell \) queries at the left-interface and only \( 2 \) queries at the right-interface.

To lower bound the restricted distinguishing advantage of the distin-
guisher $D_{\ell,n}^{\text{iter}}$, it will be convenient to consider a variation of the random oracle, denoted $\mathbf{T}$, which answers to the first $2^n$ fresh queries by a $n$-bit string chosen uniformly at random but without replacement, i.e., on the first $2^n$ fresh queries $\mathbf{T}$ corresponds to a random injective function. Trivially, $\mathbf{RO}$ and $\mathbf{T}$ are indistinguishable unless a collision occurs in $\mathbf{RO}$, i.e.,

$$\widehat{\Delta} \left( L|\mathbf{RO}|^R, L|\mathbf{T}|^R \right) \leq p_{\text{coll}} (L + R, 2^n),$$

(5.1)

for all $L, R \in \mathbb{N}$, and where recall from Section 2.1 that $p_{\text{coll}}(q, t)$ denotes the probability that there exists a collision among $q$ values distributed independently and uniformly at random over a set of $t$ elements. Let $\mathbf{R}'$ and $\mathbf{S}'$ denote the systems $\mathbf{R}$ and $\mathbf{S}$, respectively, where the random oracle $\mathbf{RO}$ was replaced by its variation $\mathbf{T}$, i.e., $\mathbf{R}' := \text{iter}_c 2c\ell|\mathbf{T}|^2$ and $\mathbf{S}' := 2\ell|\mathbf{T}|^n \sigma$. Note that (5.1) implies that $\widehat{\Delta} (\mathbf{R}', \mathbf{R}) \leq p_{\text{coll}} (2c\ell + 2, 2^n)$, and similarly $\widehat{\Delta} (\mathbf{S}', \mathbf{S}) \leq p_{\text{coll}} (2\ell + n, 2^n)$. It suffices thus to lower bound the distinguishing advantage of $D_{\ell,n}^{\text{iter}}$ when interacting with $\mathbf{R}'$ or $\mathbf{S}'$ since

$$\widehat{\Delta} (\mathbf{R}, \mathbf{S}) \geq \widehat{\Delta} (\mathbf{R}', \mathbf{S}') - (p_{\text{coll}} (2c\ell + 2, 2^n) + p_{\text{coll}} (2\ell + n, 2^n)).$$

In the random experiment defined by the distinguisher $D_{\ell,n}^{\text{iter}}$ interacting with the system $\mathbf{R}'$, we always have $y^{(c\ell+1)} = \tilde{y}^{(c\ell+1)}$. Furthermore, note that since the start of the chain $y^{(0)}$ is an $(n + 1)$-bit string, none of the elements in the two chains $\mathcal{C}$ and $\tilde{\mathcal{C}}$, which both contain $n$-bit strings coming only from the system $\mathbf{R}'$, can be the value $y^{(0)}$, and since no collision can occur in $\mathbf{R}'$ it follows that $|\tilde{\mathcal{C}}| = \ell$ and $\mathcal{C} \cap \tilde{\mathcal{C}} = \emptyset$. Thus, $D_{\ell,n}^{\text{iter}}$ always outputs 1 when interacting with $\mathbf{R}'$.

In contrast, in the random experiment defined by the distinguisher $D_{\ell,n}^{\text{iter}}$ interacting with $\mathbf{S}'$, $D_{\ell,n}^{\text{iter}}$ outputs 1 if the simulator $\sigma$ in $\mathbf{S}'$ outputs the start and the end of an $\ell$-chain $(\tilde{y}^{(1)}, \tilde{y}^{(c+1)}, \ldots, \tilde{y}^{(c(\ell-1)+1)}, \tilde{y}^{(c\ell+1)})$ of hashes later queried by $D_{\ell,n}^{\text{iter}}$, i.e., the simulator $\sigma$ must output $y^{(c\ell+1)}$ and $y^{(1)}$ such that $\tilde{y}^{(1)} = y^{(1)}$ and $\tilde{y}^{(c\ell+1)} = y^{(c\ell+1)}$. The simulator $\sigma$ does at most $n$ queries to the underlying variant of the random oracle $\mathbf{T}$ and therefore $\sigma$ receives at most $n$ responses from $\mathbf{T}$, where by assumption $n \leq \ell - 1$. Thus, at least one of the chain element $\tilde{y}^{(cj+1)}$ that the distinguisher $D_{\ell,n}^{\text{iter}}$ obtained was never received by the simulator $\sigma$, for some $j \in \{1, \ldots, \ell\}$. If $j = \ell$, then the simulator $\sigma$ never queried the second to last chain element $\tilde{y}^{(c(\ell-1)+1)}$ and therefore the last chain element $\tilde{y}^{(c\ell+1)}$ is uniformly distributed over a set of size at least $2^n - 3\ell$, since at most $2\ell + n \leq 3\ell$ queries were done to $\mathbf{T}$, so that the probability that $y^{(c\ell+1)} = \tilde{y}^{(c\ell+1)}$ is at most $1/(2^n - 3\ell)$. Similarly, if $j < \ell$ and the simulator $\sigma$ did receive all
the next chain elements $\tilde{y}^{(e(j+1)+1)}, \ldots, \tilde{y}^{(e\ell+1)}$, then $\tilde{y}^{(e(j+1)+1)}$ is uniformly distributed over a set of size at least $2^n - 3\ell$, and the probability that it equals the value input by the simulator $\sigma$ when it received the next chain element $\tilde{y}^{(e(j+1)+1)}$ is therefore at most $1/(2^n - 3\ell)$. Thus, when interacting with $S'$, the distinguisher $D_{\text{iter}}^{\ell,n}$ outputs 1 with probability at most $1/(2^n - 3\ell)$, and overall we have

$$\hat{\Delta}(R', S') \geq 1 - \frac{1}{2^n - 3\ell}.$$ 

The proof is finished by combining the two previous equations and by using the standard argument mentioned in Section 2.1 that $p_{\text{coll}}(q,t) \leq q^2/(2t)$.

### 5.2.2 An Example of a Vulnerable Application

There are concrete applications where the fact that the plain iteration protocol $\text{iter}_c$ fails to be a query-complexity amplifier is problematic. One example of such a vulnerable application is the setting of mutual proofs of work, introduced by Dodis et al. [DRST12], which is secure if a monolithic random oracle $\text{RO}$ is employed, but becomes insecure if the $c$-iterate $\text{iter}_c \text{RO}$ is used instead, for any $c \geq 2$. This fact was already known for the special case $c = 2$ [DRST12] and we will briefly describe how it generalizes to higher iteration counts.

Recall that in mutual proofs of work, two parties aim at proving to each other that they did a certain amount of computation. In the protocol proposed by Dodis et al. [DRST12], both parties exchange in the first round a nonce and then compute a chain of hashes of a certain length (chosen by the computing party) starting with the received nonce. In the second round, both parties exchange the length and the last element of their computed chain. Then, each party checks that the other party actually did the claimed amount of computation by first computing a chain of hashes of the asserted length starting with the nonce that was originally sent, and second, by checking that both computed chains do not have any common element.

Note that such a scheme is insecure if the parties use $\text{iter}_c \text{RO}$ to compute their chain of hashes. Indeed, a malicious party, similarly to the distinguisher $D_{\text{iter}}^{\ell,n}$ in System 5.3, can simply “shift” the chain of hashes computed by the honest party and needs therefore only a constant number of hash evaluations to compute the beginning and the end of a valid chain of hashes of arbitrary length (which with overwhelming probability has no common element with the chain computed by the honest party). In
contrast, this protocol for mutual proofs of work is secure if the parties use a query-complexity amplifier, such as the collision-free iteration protocol $\text{amp}_c$ presented in the next section, to compute their chain of hashes.

## 5.3 The Collision-Free Iteration Amplifier

The main result of this section is to present the collision-free iteration protocol, denoted $\text{amp}_c$, for amplifying the query complexity of a random oracle by a constant factor $c$, for some fixed parameter $c \in \mathbb{N}$. We present the (uniform) protocol $\text{amp}_c$ and the corresponding (uniform) simulator $\sigma_{\text{amp}}$ in Section 5.3.1 and prove the actual construction stated below in Section 5.3.2. We will show in Section 5.3.3 that a similar statement can also be obtained for multiple random oracles which are globally restricted.

### Theorem 5.2.

The collision-free iteration protocol $\text{amp}_c$ described in Figure 5.2 is an $(L, R) \mapsto ([L_c], [R_c]), \delta$-query-complexity amplifier with respect to the simulator $\sigma_{\text{amp}}$ detailed in System 5.4, i.e.,

$$\{ RO^L \}_{L, R \in \mathbb{N}} \xrightarrow{\text{IND}} \{ RO^L \}_{L, R \in \mathbb{N}},$$

where $\delta_{L, R} (D) := R \cdot 2^{-n}$ and $n$ is the output length of the random oracle $RO$, for all $L, R \in \mathbb{N}$ and all distinguishers $D$.

Notice that the upper bound $\delta$ on the distinguishing advantage in the previous theorem is independent of the number $L$ of queries made to the left-interface and also of the factor $c$, which also corresponds to the number of iterations in the protocol $\text{amp}_c$ given in Figure 5.2 above. Throughout this section, we will denote by $\ell$ and $\eta$ the two integers corresponding to $[L_c]$ and $[R_c]$, respectively, for all $L, R \in \mathbb{N}$.

### 5.3.1 The Protocol and the Simulator

#### Protocol $\text{amp}_c$.

Consider the collision-free iteration protocol $\text{amp}_c$ attached to the left-interface of a random oracle as described in Figure 5.2. When queried on an input $x \in \{0, 1\}^*$ (at its outside interface), the protocol $\text{amp}_c$ does $c$ queries $V_1 (x), \ldots, V_c (x)$ to the random oracle, where the query $V_j (x)$ contains the answer of the random oracle on the previous query $V_{j-1} (x)$. In addition, $\text{amp}_c$ uses prefixing to ensure that there is no collision among the queries asked, i.e., $V_j (x) \neq V_{j'} (x')$ whenever $(j, x) \neq (j', x')$. Namely, $\text{amp}_c$ prefixes each query $V_j (x)$ with a prefix-free encoding $[x]$ of $x$ and with an iteration counter $\langle j \rangle$ where $\langle \cdot \rangle : \{1, \ldots, c\} \to \{0, 1\}^{\lceil \log_2 c \rceil}$.
5.3 The Collision-Free Iteration Amplifier

\[ V_1(x) := |x| \langle 1 \rangle \]
\[ W_1(x) := V_{c-1}(x) \]
\[ V_c(x) := |x| \langle c \rangle \parallel W_{c-1}(x) \]

\textbf{Figure 5.2.} Protocol \( \text{amp}_c \) for amplifying the complexity of a random oracle by a factor \( c \). A prefix-free encoding of a bit-string \( x \) is denoted by \(|x|\), and an encoding of an integer \( j \) over \( \lceil \log_2 c \rceil \)-bit strings by \( \langle j \rangle \).

denotes an arbitrary injective function from \{1, \ldots, c\} to the set of \( \lceil \log_2 c \rceil \)-bit strings. The former guarantees no overlap between the queries for two different inputs \( x \) and \( x' \), while the second prevents collisions within the sequence of queries for the same input \( x \). More generally, letting \( W_0(x) \) be the empty bit string and \( W_j(x) \) be the inner response of the connected resource to the inner query \( V_j(x) \), we then define iteratively

\[ V_j(x) := |x| \langle j \rangle \parallel W_{j-1}(x) \]
\[ W_j(x) := \text{result of querying } V_j(x) \text{ at the in-interface,} \]

for all \( j \in \{1, \ldots, c\} \). Finally, we simply let \( W_c(x) \), the response of the connected resource to the final query, be the output of the protocol.

\textbf{Prefix-free encodings.} A prefix-free encoding function \( \lfloor \cdot \rfloor : \{0,1\}^* \rightarrow \{0,1\}^* \) is a function ensuring that \( \lfloor \tilde{x} \rfloor \) is not a prefix of \( |x| \) whenever \( x \neq \tilde{x} \). We also assume it can be easily decided whether a bit string \( y \in \{0,1\}^* \) is in the range of \( \lfloor \cdot \rfloor \), and in that case the (unique) pre-image of \( y \) can be efficiently recovered. Our results are independent of which such prefix-free encoding function is used. A simple example of such a prefix-free encoding is the function \( \lfloor \cdot \rfloor : \{0,1\}^* \rightarrow \{0,1\}^* ; (b_1, \ldots, b_n) \mapsto (1, b_1, 1, b_2, \ldots, 1, b_n, 0) \). Many other (more efficient) examples exist, such as those described by Coron et al. [CDMP05].

\textbf{Simulator} \( \sigma_{\text{amp}} \). Before describing the behavior of the simulator \( \sigma_{\text{amp}} \) defined in System 5.4, let us first characterize more precisely the different types of queries we will consider. A query \( v \) is said to be well-formed, denoted \( \text{isWellFormed}(v) \), if it contains the prefixes as used by the protocol.
$\text{amp}_c$, i.e., $v \in \mathcal{V} \subseteq \{0,1\}^*$, where $\mathcal{V} := \bigcup_{x \in \{0,1\}^*} \mathcal{V}_j(x)$, with

$$\mathcal{V}_1(x) := \{[x] \parallel \langle 1 \rangle\} \quad \text{and} \quad \mathcal{V}_j(x) := \{[x] \parallel \langle j \rangle \parallel z : z \in \{0,1\}^n\},$$

for all $j \geq 2$. An element of $\mathcal{V}_j(x)$ will be called a well-formed query of level $j$ with prefix $x$. We denote by $\text{parse}(\cdot) : \mathcal{V} \to \{0,1\}^* \times \{1,\ldots,c\}$ the function which, given a well-formed query $v$, returns the pair $(x,j)$ corresponding to the prefix and level associated with this query, respectively. Given an arbitrary subset of well-formed queries $\tilde{\mathcal{V}} \subseteq \mathcal{V}$, a prefix $x \in \{0,1\}^*$ is declared to be “fresh”, denoted $\text{isPrefixFresh}(x,\tilde{\mathcal{V}})$, if it was never encountered, i.e.,

$$\text{isPrefixFresh}(x,\tilde{\mathcal{V}}) :\iff \forall v \in \tilde{\mathcal{V}} \forall j \in \{1,\ldots,c\} : (x,j) \neq \text{parse}(v).$$

The simulator $\sigma_{\text{amp}}$ works as follows: whenever it receives a well-formed query $v \in \{0,1\}^*$ of some level $j \in \{1,\ldots,c\}$ with a “fresh” prefix $x \in \{0,1\}^*$, it emulates the behavior of the protocol $\text{amp}$ on input $x$ by generating a “fake” chain of queries $\tilde{V}_1(x), \tilde{W}_1(x), \ldots, \tilde{V}_{c-1}(x), \tilde{W}_{c-1}(x), \tilde{V}_c(x)$, where the emulated answers $\tilde{W}_k(x)$ are simply uniform $n$-bit strings locally sampled by the simulator. Then, the simulator $\sigma_{\text{amp}}$ returns the answer of the random oracle $\mathcal{RO}^{\tilde{\mathcal{V}}}$ when queried on the prefix $x$, only if the outer query $v$ matches the last chain element $\tilde{V}_c(x)$ and all previous chain elements $\tilde{V}_1(x), \ldots, \tilde{V}_{c-1}(x)$ were already queried. On the other hand, if the query $v$ matches one of the lower-level chain elements, i.e., $v = \tilde{V}_j(x)$ with $j < c$ and all previous chain elements were already queried, then the simulator $\sigma_{\text{amp}}$ replies with the answer $\tilde{W}_j(x)$ that was already chosen earlier (when generating the chain for the prefix $x$). In the (unlikely) case where a distinguisher happens to have guessed the value of $\tilde{V}_j(x)$, i.e., $v = \tilde{V}_j(x)$ but the previous chain element $\tilde{V}_{j-1}(x)$ was never queried, the simulator $\sigma_{\text{amp}}$ gives up on simulation by outputting the all zero bit string $0^n$ and setting internally the event hit to 1 in order to prevent any further inner query to the random oracle. Finally, if the query $v$ considered is not well-formed, then the simulator $\sigma_{\text{amp}}$ replies with a fresh uniform $n$-bit string. We refer to System 5.4 for a precise description of the simulator $\sigma_{\text{amp}}$. Note that it maintains a state over all invocations, keeping track of the set $\tilde{\mathcal{V}}$ of well-formed queries received, the values $\tilde{V}_j(x)$ and $\tilde{W}_j(x)$ corresponding to the locally generated chains of queries, and the mapping $g$ to be able to reply consistently to any repeated query.
5.3 The Collision-Free Iteration Amplifier

System 5.4: Simulator $\sigma_{\text{amp}}$

\[
g(v) := \diamond, \text{ for all } v \in \{0, 1\}^* \quad \tilde{V} = \emptyset \quad \text{and } \text{hit} := 0
\]

**on input** $v \in \{0, 1\}^*$ at the out-interface

if $g(v) = \diamond$ then // $v$ was never queried before

if isWellFormed($v$) then

(x, j) := parse($v$)

if isPrefixFresh($x, \tilde{V}$) then // generate a “fake” chain of queries

($\tilde{V}_1(x), \tilde{W}_1(x), \ldots, \tilde{V}_{c-1}(x), \tilde{W}_{c-1}(x)) :=$

GenerateChain($x, c - 1$)

$\tilde{V}_c(x) := \lfloor x \rceil \| \langle c \rangle \| \tilde{W}_{c-1}(x)$

else if $v = \tilde{V}_j(x)$ then // previous chain element was not queried

if $j > 1 \land g(\tilde{V}_{j-1}(x)) = \diamond$ then

hit := 1 and $\tilde{Y} := 0^n$

else if $j = c \land \text{hit} = 0$ then

$\tilde{Y} := \text{result of querying } x \text{ at the in-interface}$

else if $j = c \land \text{hit} = 1$ then

$\tilde{Y} := 0^n$

else $\tilde{Y} := \tilde{W}_j(x)$

else $\tilde{Y} \leftarrow \$ \{0, 1\}^n$

$\tilde{V} \leftarrow \tilde{V} \cup \{v\}$

else $\tilde{Y} \leftarrow \$ \{0, 1\}^n$

$g(v) := \tilde{Y}$

**output** $g(v)$ at the out-interface

Procedure GenerateChain($x, m$)

$\tilde{W}_0(x) := \diamond$

for $j = 1$ to $m$ do

$\tilde{V}_k(x) := \lfloor x \rceil \| \langle k \rangle \| \tilde{W}_{k-1}(x)$

$\tilde{W}_k(x) \leftarrow \$ \{0, 1\}^n$

return $(\tilde{V}_1(x), \tilde{W}_1(x), \ldots, \tilde{V}_m(x), \tilde{W}_m(x))$

5.3.2 Indistinguishability Proof

In this section we prove that the collision-free iteration protocol $\text{amp}_c$ is a $c$-fold query-complexity amplifier as stated in Theorem 5.2. The proof consists of the following two parts: First, we show in Lemma 5.3 below that
the game \( \overline{\text{RO} \sigma_{\text{amp}}} \), which is defined to be won if the event hit is provoked in the simulator \( \sigma_{\text{amp}} \) described in System 5.4, is conditionally equivalent to the system \( \text{amp}_c \text{RO} \). Second, we use Lemma 4.3 in Section 4.3.1 to derive the desired construction.

**Lemma 5.3.** Consider the protocol \( \text{amp}_c \) and the simulator \( \sigma_{\text{amp}} \) defined in Figure 5.2 and System 5.4, respectively. Let \( \overline{\text{RO} \sigma_{\text{amp}}} \) denote the game which is won if and only if the event hit in \( \sigma_{\text{amp}} \) is provoked. Then,

\[
\overline{\text{RO} \sigma_{\text{amp}}} \equiv \text{amp}_c \text{RO} \quad \text{and} \quad \hat{\Gamma}^{\text{NA}} \left( L_{\overline{\text{RO} \sigma_{\text{amp}}}}^R \right) \leq R \cdot 2^{-n},
\]

for all \( L, R \in \mathbb{N} \).

**Proof.** Let us denote by \( \mathbf{R} \) and \( \mathbf{S} \) the systems \( \text{amp}_c \text{RO} \) and \( \text{RO} \sigma_{\text{amp}} \), respectively. Let \( \mathbf{S} \) be the game which is won if and only if the event hit in \( \sigma_{\text{amp}} \) is provoked. We first show that the game \( \mathbf{S} \) is conditionally equivalent to the system \( \mathbf{R} \) and later analyze the probability of winning the query-restricted game \( \overline{L_{\mathbf{S}}^R} \).

We are going to argue that as long as the game \( \mathbf{S} \) is not won, the probability distribution of the response to any possible query is the same in both \( \mathbf{R} \) and \( \mathbf{S}^{-1} \). Both systems reply consistently to any repeated queries, let us hence without loss of generality only consider fresh queries. To analyze the sampling process of responses, note that we can see both \( \mathbf{R} \) and \( \mathbf{S}^{-1} \) as generating the responses to all possible queries in advance (according to distributions described below) and then using the pre-generated responses to answer all actual queries. To describe these distributions, let us denote by Left \((x)\) and Right \((v)\) the responses of the system in question (either \( \mathbf{R} \) or \( \mathbf{S}^{-1} \)) to queries \((\text{left}, x)\) and \((\text{right}, v)\), respectively. The (inefficient) sampling processes for the systems \( \mathbf{R} \) and \( \mathbf{S}^{-1} \) (as long as the game is not won) are described in Systems 5.5 and 5.6, respectively. It is now easy to see that these two sampling processes result in the same distribution of all the random variables Left \((x)\) and Right \((v)\).

We now upper bound the probability for some non-adaptive distinguisher \( \mathbf{D} \) to win the query-restricted game \( \overline{L_{\mathbf{S}}^R} \), for some integers \( L \) and \( R \). The game winner \( \mathbf{D} \) wins the game only if it provokes the event hit in the simulator \( \sigma_{\text{amp}} \) within \( R \) (well-formed) queries. Any well-formed query has a certain prefix \( x \) and level \( j \), where \( x \in \{0, 1\}^* \) and \( j \in \{1, \ldots, c\} \), and the probability for such a query to win the game is therefore at most \( 2^{-n} \) since it requires to guess the value of \( \tilde{W}_{j-1}(x) \), an independent and
System 5.5: Sampling for $R$

1. foreach $x \in \{0, 1\}^*$ do
   \[ U_1, \ldots, U_c \leftarrow \{0, 1\}^n \]
   \[ \text{Right} ([x] \parallel \langle 1 \rangle) := U_1 \]
   \[ \text{Right} ([x] \parallel \langle j \rangle \parallel U_{j-1}) := U_j, \]
   for all $j \in \{1, \ldots, c\}$
   \[ \text{Left} (x) := U_c \]
2. Sample all remaining values
   \[ \text{Right} (v) \leftarrow \{0, 1\}^n \]

System 5.6: Sampling for $S$^{-1}$

1. foreach $x \in \{0, 1\}^*$ do
   \[ \text{Left} (x) \leftarrow \{0, 1\}^n \]
   \[ U_1, \ldots, U_{c-1} \leftarrow \{0, 1\}^n \]
   \[ \text{Right} ([x] \parallel \langle 1 \rangle) := U_1 \]
   \[ \text{Right} ([x] \parallel \langle j \rangle \parallel U_{j-1}) := U_j, \]
   for all $j \in \{1, \ldots, c-1\}$
   \[ \text{Right} ([x] \parallel \langle c \rangle \parallel U_{c-1}) := \text{Left} (x) \]
2. Sample all remaining values
   \[ \text{Right} (v) \leftarrow \{0, 1\}^n \]

uniformly distributed $n$-bit string. By applying the union bound it follows that $\hat{\Gamma}^{\text{NA}}_{R \sigma_{\text{amp}} L} \leq R \cdot 2^{-n}$, for all $L, R \in \mathbb{N}$.

In order to use Lemma 4.3, we first need to argue about the query complexity of the protocol $\text{amp}_c$ and of the simulator $\sigma_{\text{amp}}$. Recall from Definition 4.2 that $\text{QC}_\alpha (L)$ denotes the maximum number of inner queries made by the converter $\alpha$ for any $L$ outer queries.

Query complexity. The protocol $\text{amp}_c$ makes exactly $c$ inner queries for every query it receives at its outside interface. Consequently, the protocol $\text{amp}_c$ does in total at most $L$ inner queries if it is queried at most $\ell$ times at its outside interface. The simulator $\sigma_{\text{amp}}$ makes a query $x$ at its inside interface only if it receives a chain of $c$ (distinct) queries $\tilde{V}_1 (x), \ldots, \tilde{V}_c (x)$. The simulator $\sigma_{\text{amp}}$ keeps in memory the previous interaction, so that when such a chain of $c$ queries is received, at most one query is made to the inside interface of $\sigma_{\text{amp}}$. Furthermore, the prefix scheme employed prevents any form of collision among the queries so that making multiple, say $k$, such chains of $c$ queries requires at least $k \cdot c$ queries. Hence, any tuple of $R$ queries contains at most $n$ such chains of $c$ queries, and thus the simulator does in total at most $n$ inner queries if it is queried at most $R$ times at its outside interface. Thus, the protocol $\text{amp}_c$ and the simulator $\sigma_{\text{amp}}$ are such that

\[ \text{QC}_{\text{amp}_c} (L) \leq c \cdot L \quad \text{and} \quad \text{QC}_{\sigma_{\text{amp}}} (R) \leq n, \]

for all integers $L$ and $R$.

The conditional equivalence statement shown in Lemma 5.3 together with Lemma 4.3 imply then that the uniform family of protocols $\pi :=$
The uniform family of simulators \( \sigma := \{ \sigma_{\text{amp}|R} \}_{L,R \in \mathbb{N}} \) are such that

\[
\{ cL|RO |R \}_{L,R \in \mathbb{N}} \xrightarrow{\text{IND}} \{ L|RO |R \}_{L,R \in \mathbb{N}},
\]

where \( \varepsilon_{L,R}(D) \leq \hat{\Gamma}_{\text{NA}}(L|RO \sigma_{\text{amp}}|R) \) for all \( L, R \in \mathbb{N} \) and all distinguishers \( D \). Note that the previous family of constructions hold for any integer \( L \) and so in particular also hold for all integers of the form \( \lfloor \frac{L}{c} \rfloor = \ell \), which leads to the desired family of constructions. The proof is then finished by upper bounding the probability of winning the game \( c|RO \sigma_{\text{amp}}|R \) for non-adaptive distinguishers as given in Lemma 5.3.

### 5.3.3 Globally-Restricted Random Oracles

In this section we show that the collision-free iteration protocol \( \text{amp}_c \) can also be used in parallel to decrease the number of adversarial queries that can be made to multiple random oracles which are **globally** restricted.

**Assumed resource.** The parameterized family of constructions shown below assume for each value of the parameter \( (L, R) \in \mathbb{N} \times \mathbb{N} \) the availability of \( r \) random oracles \( L|[RO, \ldots, RO]|R \) which are **globally restricted**. These random oracles can for example be constructed by applying standard domain separation techniques, such as the salting technique described in Section 3.4.4, on a single query-restricted random oracle \( L|RO |R \).

**Protocol.** The collision-free iteration protocol \( \text{amp}_c \) depicted in Figure 5.2 is applied to each random oracle \( RO \) in \( L|[RO, \ldots, RO]|R \). Overall, the protocol considered consists of taking the \( r \)-parallel composition of the protocol \( \text{amp}_c \) denoted \( \text{amp}_c^r \), where \( \text{amp}_c^r := \langle \text{amp}_c, \ldots, \text{amp}_c \rangle \).

**Desired resource.** Similarly to Definition 5.1, we wish to construct \( r \) globally restricted random oracles \( \ell|[RO, \ldots, RO]|r \) where the number of adversarial queries decreased, i.e., \( r < R \).

The next theorem shows that applying locally and independently the collision-free iteration protocol \( \text{amp}_c \) on each assumed random oracle will globally reduced the number of adversarial queries by a factor \( c \). As explained in Section 4.3.2, this kind of statements does in general not follow from the fact that the construction notion composes in parallel (in the sense of Theorem 2.1) since the assumed random oracles are not
The Collision-Free Iteration Amplifier 121

independent due to their global constraints. The high level idea to prove the theorem below is to use instead Lemma 4.4 to leverage the (stronger) conditional equivalence statement shown in Lemma 5.3.

**Theorem 5.3.** Recall the collision-free iteration protocol $\text{amp}_c$ described in Figure 5.2 and the simulator $\sigma_{\text{amp}}$ detailed in System 5.4. Then, the combined protocol $\text{amp}_c^r := \langle \text{amp}_c, \ldots, \text{amp}_c \rangle$ and the combined simulator $\sigma_{\text{amp}}^r := \langle \sigma_{\text{amp}}, \ldots, \sigma_{\text{amp}} \rangle$ are such that

$$\left\{ L | RO, \ldots, RO \right\}_{L,R \in \mathbb{N}} \overset{\text{IND}}{\xrightarrow{\text{amp}_c^r, \sigma_{\text{amp}}^r, \delta}} \left\{ L | RO, \ldots, RO \right\}_{L,R \in \mathbb{N}},$$

where $\delta_{L,R}(D) := R \cdot 2^{-n}$ and $n$ is the output length of each random oracle $RO$, for all $L,R \in \mathbb{N}$ and all distinguishers $D$.

**Proof.** Let us denote by $T$ the $r$ random oracles $[RO, \ldots, RO]$ composed in parallel within the proof. Note that Lemma 5.3 gives rise to a family of $r$ identical conditional equivalence statements, and together with Lemma 4.4 we obtain the following family of parameterized constructions

$$\left\{ \text{QC}_{\text{amp}_c^r}(L) | T \right\}_{L,R \in \mathbb{N}} \overset{\text{IND}}{\xrightarrow{\text{amp}_c^r, \sigma_{\text{amp}}^r, \varepsilon}} \left\{ L | T | \text{QC}_{\sigma_{\text{amp}}^r}(R) \right\}_{L,R \in \mathbb{N}},$$

where $\varepsilon_{L,R}(D) \leq \sum_{i=1}^{r} \text{QC}_{\sigma_{\text{amp}}^r}(R_i)$, for all $L,R \in \mathbb{N}$ and all distinguishers $D$. All that remains to finish the proof of Theorem 5.3 is to upper bound the query complexity of the protocol $\text{amp}_c^r$ and of the simulator $\sigma_{\text{amp}}^r$, and to upper bound the probability of winning the disjunction of $r$ independent instances of the game $RO \sigma_{\text{amp}}$ in a non-adaptive manner.

**Query complexity.** Let us first upper bound the query complexity of the simulator $\sigma_{\text{amp}}^r$. Recall from Section 5.3.1 that the simulator $\sigma_{\text{amp}}$ does at most $\left\lfloor \frac{R}{c} \right\rfloor$ inner queries for every tuple of $R$ outside queries it receives. Then, Lemma 4.6 implies that for all $R \in \mathbb{N}$

$$\text{QC}_{\sigma_{\text{amp}}^r}(R) \leq \max \text{QC}_{\sigma_{\text{amp}}}(R_1) + \cdots + \text{QC}_{\sigma_{\text{amp}}}(R_r),$$

$$= \max \left\lfloor \frac{R_1}{c} \right\rfloor + \cdots + \left\lfloor \frac{R_r}{c} \right\rfloor,$$

$$\leq \max \left\lfloor \frac{R_1 + \cdots + R_r}{c} \right\rfloor,$$

$$= \left\lfloor \frac{R}{c} \right\rfloor.$$
where the maximum is taken over all integers \( R_1, \ldots, R_r \) summing up to \( R \) and the second inequality comes from the fact that the function \( a \mapsto \lfloor \frac{a}{c} \rfloor \) is superadditive\(^3\) over non-negative reals. Therefore, the combined simulator \( \sigma^{\text{amp}}_r \) has the same query complexity as the single simulator \( \sigma^{\text{amp}} \).

The query complexity of the combined protocol \( \text{amp}^{r}_{c} \) can be upper bounded in a similar manner. Namely, recall from Section 5.3.1 that the query complexity of the single protocol \( \text{amp}^{c}_{c} \) is a linear function. Therefore, Lemma 4.6 directly implies that

\[
\text{QC}_r(L) \leq L \cdot c,
\]

for all \( L \in \mathbb{N} \), and thus the combined protocol \( \text{amp}^{r}_{c} \) has the same query complexity as the protocol \( \text{amp}^{c}_{c} \) and does in particular at most \( L \) inner queries if it receives \( \lfloor \frac{L}{c} \rfloor \) outer queries.

**Game winning.** The last part of the proof consists of upper bounding the probability of winning in a non-adaptive manner a restricted game formed by the disjunction of \( r \) identical games \( \text{RO} \sigma^{\text{amp}} \). Recall from Lemma 5.3 that the probability of winning in a non-adaptive manner a single instance of the query-restricted game \( L \rightarrow \text{RO} \sigma^{\text{amp}} |^R \) is at most \( R \cdot 2^{-n} \), which is a linear function of \( R \). Thus, Lemma 4.5 implies

\[
\hat{\Gamma}^{\text{NA}}(L | [\text{RO} \sigma^{\text{amp}} |^R] \ldots, [\text{RO} \sigma^{\text{amp}} |^R]) \leq R \cdot 2^{-n},
\]

for all integers \( L \) and \( R \).

\[ \square \]

### 5.4 Towards Optimality

Theorem 5.2 gives a simple protocol to amplify the query-complexity of a random oracle by a constant factor \( c \), at the cost of simultaneously reducing the number of honest-party queries by the same factor. Such a decrease is of course undesired, and the goal of this section is to study whether such a reduction of the honest-party queries is inherent to any query-complexity amplification scheme. To do so, we consider an arbitrary (deterministic) protocol \( \pi \) and an arbitrary simulator \( \sigma \), and show in Lemma 5.4 that in order for the construction \( L \rightarrow \text{RO} |^R (\pi, \sigma, \varepsilon) \rightarrow \text{IND} |^n \) to be achievable, the composed converter \( \sigma \circ \pi \) must do (with high probability) at least as many inner queries as many outer queries it receives. This in particular implies

\[ \text{A function } f : \mathcal{A} \rightarrow \mathbb{R}, \text{ where } \mathcal{A} \subseteq \mathbb{R} \text{ and } \mathcal{A} \text{ is closed under addition, is said to be superadditive if } f(a) + f(b) \leq f(a + b), \text{ for all } a, b \in \mathcal{A}. \]

\[ \text{ }\]
5.4 Towards Optimality

System 5.7: Distinguisher $D_{\pi, x^k}^{QCA}$ interacting with $S$

\[
\begin{align*}
B & := 0 \\
\text{for } j = 1 \text{ to } k \text{ do } & \quad y_j := \text{result of querying } x_j \text{ at the left-interface of } S \\
\text{for } j = 1 \text{ to } k \text{ do } & \quad \tilde{y}_j := \text{EmulateProtocol(right, } x_j) // \text{ Emulate } \pi \text{ at right} \\
\text{if } \exists j \in \{1, \ldots, k\} : y_j \neq \bullet \land y_j \neq \tilde{y}_j \text{ then } & \quad B := 1 \\
\text{return } B \\
\end{align*}
\]

Procedure EmulateProtocol($i, x$)

\[
\begin{align*}
(i', x') := & \quad \text{emulate result of querying } \pi \text{ at the out-interface on } x \\
\text{while } i' = \text{in } do & \\
\quad y' := & \quad \text{result of querying } x' \text{ at the } i\text{-interface of } S \\
(i', x') := & \quad \text{emulate result of querying } \pi \text{ at the in-interface on } y' \\
\text{return } x'
\end{align*}
\]

that if the simulator $\sigma$ considered is such that for every $n$-tuple of outer queries it does at most $n/\ell$ inner queries, for some positive constant $c$, then the protocol $\pi$ must do at least $c \cdot \ell$ inner queries for every $\ell$-tuple of outer queries it receives. Therefore, the protocol $\text{amp}_c$ given in Theorem 5.2 is in this sense optimal (if one restricts oneself to such a class of simulators). In the following, we denote by $Q_\alpha(x^k)$ the random variable corresponding to the sequence of inner queries made by $\alpha$ when queried at its outside interface on $x_1, \ldots, x_k$ in a given random experiment (e.g., a distinguisher interacting with $RO\alpha$). The number of distinct elements in such a tuple, which is also a random variable, will be denoted by $|Q_\alpha(x^k)|$.

Lemma 5.4. Consider four integers $L, R, \ell, n \in \mathbb{N}$, where $R \geq L > 0$ and $\ell > 0$. Let $\pi \in \Sigma$ be a deterministic protocol and let $\sigma \in \Sigma$ be a simulator. Then, for any integer $k \leq \ell$ and for any sequence of $k$ distinct bit strings $x^k = (x_1, \ldots, x_k) \in \{(0, 1)^*\}^k$, there exists a distinguisher $D_{\pi, x^k}^{QCA}$ described in System 5.7 such that

\[
\tilde{\Delta}_{\sigma}^{D_{\pi, x^k}^{QCA}} (\pi^{L \cdot RO\lceil R \rceil}, \ell \cdot RO^{\lceil n \rceil} \sigma) \geq (1 - 2^{-n}) \cdot P (|Q_\sigma (Q_\pi (x^k))| < k) ,
\]

where the probability is taken in the random experiment defined by $D_{\pi, x^k}^{QCA}$ interacting with the system $\left[\ell \cdot RO^{\lceil n \rceil} \sigma\right]$. 

Proof. Consider an integer $k$ such that $k \leq \ell$ and a sequence of $k$ distinct bit strings $x_1, \ldots, x_k$, where $x_j \in \{0, 1\}^*$. Consider the distinguisher $D_{\pi, x^k}^{QCA}$ described in System 5.7. In the $D_{\pi, x^k}^{QCA} S$ random experiment, where $S$
is one of the two systems $\left[ \pi^{\ell}\mathsf{RO}^{R} \right]^{-1}$ or $\left[ \ell\mathsf{RO}^{R} \sigma \right]^{-1}$, the distinguisher $D_{\pi,x^k}^{\text{QCA}}$ emulates the protocol $\pi$ at the right-interface of $S$. The distinguisher $D_{\pi,x^k}^{\text{QCA}}$ first queries the left-interface of $S$ on the sequence of inputs $x^k$, obtaining the corresponding sequence of answers $y^k$, and queries also the right-interface of $S$ (through the emulated protocol $\pi$) on the same sequence of inputs $x^k$ to obtain the sequence of responses $\tilde{y}^k$. The distinguisher outputs its decision bit $B$ which is 1 if and only the answers obtained are not consistent, i.e., $y_j \neq \bullet$ and $y_j \neq \tilde{y}_j$ for some $j \in \{1, \ldots, k\}$.

Consider first the random experiment where the distinguisher $D_{\pi,x^k}^{\text{QCA}}$ interacts with the system $\left[ \pi^{\ell}\mathsf{RO}^{R} \right]^{-1}$. If the answer $y_j$ obtained at the left-interface during the $j^{th}$ query is such that $y_j \neq \bullet$, then the protocol $\pi$ did at most $L$ inner queries to the left-interface of the random oracle $\mathsf{RO}^{L\mathsf{R}}$. Since the protocol $\pi$ is assumed to be deterministic, it follows therefore that when queried on the same sequence of inputs $x^k$, the emulated protocol $\pi$ at the right-interface of the random oracle $\mathsf{RO}^{L\mathsf{R}}$ did also at most $L \leq R$ queries and thus $y_j = \tilde{y}_j$ whenever $y_j \neq \bullet$, for all $j \in \{1, \ldots, k\}$. Thus, the decision bit $B$ output by $D_{\pi,x^k}^{\text{QCA}}$ in such a random experiment can never be 1.

Consider now the random experiment where the distinguisher $D_{\pi,x^k}^{\text{QCA}}$ interacts instead with the system $\left[ \ell\mathsf{RO}^{R} \sigma \right]^{-1}$. The probability that the decision bit $B$ output by $D_{\pi,x^k}^{\text{QCA}}$ is 1 in such a random experiment is trivially lower bounded by that of the joint event $B = 1$ and $|Q_\sigma (Q_\pi (x^k))| < k$. Since the distinguisher $D_{\pi,x^k}^{\text{QCA}}$ first does $k \leq \ell$ queries to the left-interface of the random oracle $\ell\mathsf{RO}^{R\sigma}$, the answers $y^k$ to such queries are such that $y_j \neq \bullet$, for all $j \in \{1, \ldots, k\}$. Given the event $|Q_\sigma (Q_\pi (x^k))| < k$, at least one of the inputs $x_j$ was never queried by the simulator $\sigma$ and thus the corresponding answer $\tilde{y}_j$ will differ from $y_j$ with probability at least $1 - 2^{-n}$.  \[\square\]
Part II

Common Randomness between Two Distrustful Parties
Chapter 6

Common Randomness Amplification

We show in this chapter that common randomness cannot be amplified by interactive protocols between two distrustful parties. In Section 6.1, we formalize the notion of common randomness amplifiers as a construction of a certain common randomness resource. The basic constraints involved in such a construction are further studied in Section 6.2. These constraints will lay the groundwork for our impossibility result presented in Section 6.3, where for the sake of clarity we first detail the simpler case of perfect security in Section 6.3.1 and then deal with the more general case of statistical security in Section 6.3.2.

This chapter is stated in the setting of two distrustful parties described in Section 2.5.3. Resources in this chapter have therefore a left- and a right-interface, and the set of interfaces is $\mathcal{I} := \{\text{left}, \text{right}\}$. The information-theoretic tools from Section 2.2 will play a crucial role in this chapter.

6.1 Common Randomness Amplifiers

We are interested in protocols aiming at amplifying common randomness between two distrustful parties. The goal of such a protocol is for both parties to agree on a common random value $W$, given solely some initial common random value $U$ and a bi-directional communication channel to exchange messages. The term amplification refers to the fact that the
Sources of common randomness. We model the source of common randomness as an explicit resource and consider two variations thereof, denoted CRS and CR. Both resources are parameterized by a distribution $P_V$ over some set $V$. The common reference string resource $\text{CRS} (P_V)$ first samples from the distribution $P_V$ to obtain a value $v$ and then outputs $v$ at interface $i \in \{ \text{left, right} \}$ whenever it receives an input $\text{getseed}$ at the same interface $i$. In contrast, the common randomness resource $\text{CR} (P_V)$ only outputs the sampled value $v$ when queried on $\text{getseed}$ at interface $i$ if a message $dlv$ was previously input at the other interface. The resources CRS and CR are described in Systems 6.1 and 6.2, respectively, where given an interface $i \in \{ \text{left, right} \}$ we denote by $i^C$ the other interface.

Notice that the distribution of the sampled value $v$ cannot be influenced by any party in both resources $\text{CRS} (P_V)$ and $\text{CR} (P_V)$, but the latter resource allows in addition a party to decide when the other party can receive the sampled value $v$. The reason for this relaxation is that the resource $\text{CRS} (P_V)$ is trivially impossible to construct by interactive protocols since the sampled value $v$ can be requested even before the parties exchanged any messages.

Throughout this chapter, we consider two different sources of common randomness. The first is given by the resource $\text{CRS} (P_V)$, while the second is the filtered resource $\text{CR}_\psi (P_V)$, where $\psi := (\psi, \psi)$ and $\psi$ is a converter.
which behaves as follows: Initially $\psi$ outputs $dL$ at its interface in and blocks any message $\text{getseed}$ input at its interface in, while other inputs made to its in- or out-interface are directly forwarded to the other interface. The filtered resource $CR_{\psi}(P_{V})$ guarantees that when both parties are honest they both receive the same sample $v$ drawn from the distribution $P_{V}$, while a dishonest party can initially retrieve the sample $v$ and decide when the other party is able to retrieve it. Note that the systems defined are such that $CRS(P_{V}) = \psi CR(P_{V}) \psi$.

**Assumed resources.** The construction statements considered in this chapter assume two resources: an initial source of common randomness given by the common reference string resource $CRS(P_{U})$, for some distribution $P_{U}$ over a set $U$, and a bi-directional communication channel $\leftrightarrow$. The channel is parameterized by a message space $M$ and a number of messages $p$ that can be exchanged (this restriction can be formalized as a monotone binary output to obtain a restricted system in the sense of Section 2.4.3). The channel $\leftrightarrow_{M,p}$ simply forwards any message input at the left- or at the right-interface to the other interface until $p$ messages have been exchanged.

There are two main reasons why the parties are assumed to share the resource $CRS(P_{U})$ instead of the (filtered) resource $CR_{\psi}(P_{U})$. First, note that the resource $CRS(P_{U})$ is (strictly) stronger than the filtered resource $CR_{\psi}(P_{U})$. Indeed, it is easy to see that the identity protocol $(1,1)$ perfectly constructs the filtered resource $CR_{\psi}(P_{U})$ from $CRS(P_{U})$ and with respect to the simulator $(\psi,\psi)$. Consequently, any impossibility result stating that a resource $T$ cannot be constructed from the resource $CRS(P_{U})$ also implies that the same resource $T$ cannot be constructed from the filtered resource $CR_{\psi}(P_{U})$ (since otherwise the sequential composability of the construction notion would imply that $T$ can be constructed from $CRS(P_{U})$, thus leading to a contradiction). Second, assuming that the parties initially have access to $CRS(P_{U})$ will simplify our exposition and better fit the motivation that both parties initially share some amount of common randomness before initiating the protocol.

**Protocol.** We consider a two-party protocol $(\alpha, \beta)$ having access to the assumed resource $[CRS(P_{U}), \leftrightarrow_{M,p}]$. Without loss of generality, we assume that both converters first retrieve the seed $u$ from the resource $CRS(P_{U})$ and then exchange an even number $p$ of messages $M_{1}, \ldots, M_{p}$ over $M$ using the channel $\leftrightarrow_{M,p}$, where messages with odd indices are
sent by $\alpha$, while messages with even indices are sent by $\beta$.

**Desired resource.** The goal of the protocol $(\alpha, \beta)$ is to construct a filtered source of common randomness $\text{CR}_\psi (P_W)$, where the desired distribution $P_W$ contains more entropy than the initially shared amount of randomness contained in $\text{CRS} (P_U)$, i.e., $H (W) > H (U)$.

The previous considerations motivate the following definition of a common randomness amplifier.

**Definition 6.1.** Let $P_U$ and $P_W$ be two distributions such that $H (W) > H (U)$ and let $\varepsilon := (\varepsilon_1, \varepsilon_2, \varepsilon_3)$ be a triple of non-negative reals. Then, a two-party protocol $\pi := (\alpha, \beta) \in \Sigma^2$ is said to be a $(P_U, P_W, p, \varepsilon)$-common randomness amplifier with message space $M$ and with respect to a simulator $\sigma := (\nu, \tau) \in \Sigma^2$, if

$$\begin{bmatrix} \text{CRS} (P_U), \leftarrow M, p \end{bmatrix} \xrightarrow{(\pi, \sigma, \varepsilon)_{\text{TDP}}} \text{CR}_\psi (P_W).$$

### 6.2 Constructing Common Randomness

A first requirement for a $(P_U, P_W, p, \varepsilon)$-common randomness amplifier is according to Definition 6.1 to construct a (filtered) common randomness resource $\text{CR}_\psi (P_W)$. In this section, we study the basic constraints that must be satisfied for a protocol to construct such a source of common randomness. To do so, let us assume that there exist a protocol $\pi = (\alpha, \beta)$ and a simulator $\sigma = (\nu, \tau)$ such that

$$\begin{bmatrix} \text{CRS} (P_U), \leftarrow M, p \end{bmatrix} \xrightarrow{(\pi, \sigma, \varepsilon)_{\text{TDP}}} \text{CR}_\psi (P_W),$$

for some distributions $P_U$ and $P_W$. Recall from Section 2.5.3 that the construction notion in the setting of two distrustful parties and with statistical security involves the following three conditions

$$\Delta^D \left( \alpha \begin{bmatrix} \text{CRS} (P_U), \leftarrow M, p \end{bmatrix} \beta, \psi \text{CR}_\psi (P_W) \right) \leq \varepsilon_1, \quad (6.1)$$
$$\Delta^D \left( \alpha \begin{bmatrix} \text{CRS} (P_U), \leftarrow M, p \end{bmatrix}, \psi \text{CR}_\psi (P_W) \tau \right) \leq \varepsilon_2, \quad (6.2)$$
$$\Delta^D \left( \begin{bmatrix} \text{CRS} (P_U), \leftarrow M, p \end{bmatrix} \beta, \nu \text{CR}_\psi (P_W) \right) \leq \varepsilon_3, \quad (6.3)$$

for all distinguishers $D$. 
6.2 Constructing Common Randomness

Figure 6.1. The protocol \((\alpha, \beta)\) receives the same seed \(U\) and output \((Y, Z)\) after having exchanged \(p\) messages \(M_1, \ldots, M_p\) through the channel \(\overleftrightarrow{M,p}\). The random variables involved in this random experiment are distributed according to \(R_{U,M_pYZ}\). Messages \text{getseed} were omitted for clarity.

6.2.1 Availability Condition

Consider a distinguisher \(D\) trying to tell apart \(\alpha \left[ \text{CRS} (P_U), \overleftrightarrow{M,p} \right] \beta\) from \(\psi \text{CR} (P_W) \psi\) as follows. The distinguisher \(D\) inputs \text{getseed} at both the left- and right-interface to retrieve the corresponding output at each interface. When connected to the real system \(\alpha \left[ \text{CRS} (P_U), \overleftrightarrow{M,p} \right] \beta\), the distinguisher \(D\) observes the output \(Y\) of the converter \(\alpha\) at the left-interface and the output \(Z\) of the converter \(\beta\) at the right-interface. The random variables \(Y, Z\) are distributed according to some joint distribution \(R_{U,M_pYZ}\), where \(U\) is the output of \(\text{CRS} (P_U)\) and \(M_p\) are the messages exchanged between \(\alpha\) and \(\beta\). The information flow defined by the distribution \(R_{U,M_pYZ}\) is shown in Figure 6.1, where the messages \text{getseed} were omitted for simplicity.

In contrast, when the distinguisher \(D\) is connected to \(\psi \text{CR} (P_W) \psi\), it observes the same random variable \(W\) output at both interfaces. The maximum distinguishing advantage of such a distinguisher is given by the statistical distance and (6.1) implies

\[
d (R_{YZ}, P_{WW}) \leq \varepsilon_1.
\]  

(6.4)

6.2.2 Simulatability Against a Malicious Bob

We consider a specific distinguisher \(D_{\beta}\) trying to tell apart the resource \(\alpha \left[ \text{CRS} (P_U), \overleftrightarrow{M,p} \right]\) from \(\psi \text{CR} (P_W) \tau\) as follows. The distinguisher \(D_{\beta}\) emulates the converter \(\beta\) at the right-interface of the connected system and then inputs \text{getseed} at the left-interface and at the out-interface of
the emulated converter $\beta$. The distinguisher $D_\beta$ is shown in Figure 6.2. When connected to $\alpha \left[ \text{CRS} \left( P_U \right), \xrightarrow{M,p} \right]$ (see Figure 6.2a), such a distinguisher $D_\beta$ sees the random variables $U, M^p, Y, Z$ distributed according to $R_{U^pM^pYZ}$. In contrast, when connected to $\psi \text{CR} \left( P_W \right) \tau$ (see Figure 6.2b) the distinguisher sees the random variables $U, M^p, Y, Z$ distributed according to some perhaps different joint distribution, say $S_{WU^pM^pYZ}$, where $W$ denotes the random variable sampled by the common randomness resource $\text{CR} \left( P_W \right)$ which is a priori not necessarily output to the distinguisher $D_\beta$. If the simulator $\tau$ never input $d_{lv}$ to the resource $\text{CR} \left( P_W \right)$, in which case the filter $\psi$ does not output anything, we will consider $Y$ to be the error symbol $\star$.

The maximum distinguishing advantage achieved by $D_\beta$ is given by the statistical distance and (6.2) implies that

$$d \left( R_{U^pM^pYZ}, S_{U^pM^pYZ} \right) \leq \varepsilon_2 .$$

Note that in the random experiment defined by the joint distribution $S_{WU^pM^pYZ}$, messages with even indices are actually sent by the (emulated) protocol $\beta$, while the output $W$ of the resource $\text{CR} \left( P_W \right)$ is fixed from the beginning of the random experiment and is unknown to $\beta$. In other words, the distribution $S_{WU^pM^pYZ}$ is such that $W$ and a message $M_j$ sent by $\beta$ are conditionally independent given the simulated seed $U$ and the previous messages $M^{j-1}$, i.e.,

$$I^S \left( W ; M_j \mid U M^{j-1} \right) = 0, \quad \text{for all even } j \in \{1, \ldots, p\} .$$

### 6.2.3 Simulatability Against a Malicious Alice

Similarly, we consider a distinguisher $D_\alpha$ trying to tell apart the resource $\left[ \text{CRS} \left( P_U \right), \xrightarrow{M,p} \right] \beta$ from $\nu \text{CR} \left( P_W \right) \psi$ by emulating the converter $\alpha$ at the left-interface of the connected system and then by inputting $\text{getseed}$ at the out-interface of the emulated converter $\alpha$ and at the right-interface. When connected to $\left[ \text{CRS} \left( P_U \right), \xrightarrow{M,p} \right] \beta$, such a distinguisher sees the random variables $U, M^p, Y, Z$ distributed according to $R_{U^pM^pYZ}$; whereas when connected to $\nu \text{CR} \left( P_W \right) \psi$ such a distinguisher sees the random variables $U, M^p, Y, Z$ possibly distributed according to some other joint distribution, say $T_{WU^pM^pYZ}$, where $W$ is the random variable sampled by the common randomness resource $\text{CR} \left( P_W \right)$ which is a priori not necessarily output to the distinguisher $D_\alpha$. The random variable $Z$ will take on the
6.2 Constructing Common Randomness

\[ \text{CRS(} P_U \text{)} \]

\[ \mathbf{U} \]

\[ \mathbf{M}_1 \]

\[ \mathbf{M}_2 \]

\[ \mathbf{M}_{p-1} \]

\[ \mathbf{M}_p \]

\[ \mathbf{Y} \]

\[ \mathbf{Z} \]

\[ \mathbf{D}_\beta \]

(a) Real World. The distinguisher \( \mathbf{D}_\beta \) internally emulates the converter \( \beta \) at the right interface of \( \alpha \left[ \text{CRS(} P_U \text{)}, \mathbb{M}_{p,\mathcal{P}} \right] \). The distinguisher \( \mathbf{D}_\beta \) sees the random variables \( U, \mathbf{M}_p, Y, Z \) distributed according to \( R_{U|M_p Y Z} \).

\[ \text{CR(} P_W \text{)} \]

\[ \mathbf{U} \]

\[ \mathbf{M}_1 \]

\[ \mathbf{M}_2 \]

\[ \mathbf{M}_{p-1} \]

\[ \mathbf{M}_p \]

\[ \mathbf{W} \]

\[ \mathbf{Y} \]

\[ \mathbf{Z} \]

\[ \mathbf{D}_\beta \]

(b) Ideal World. The distinguisher \( \mathbf{D}_\beta \) internally emulates the converter \( \beta \) at the right interface of \( \psi \left[ \text{CR(} P_W \text{)} \right] \). The distinguisher \( \mathbf{D}_\beta \) sees the random variables \( U, \mathbf{M}_p, Y, Z \) distributed according to \( S_{W|M_p Y Z} \).

Figure 6.2. A distinguisher can internally emulate the converter \( \beta \) in order to distinguish \( \alpha \left[ \text{CRS(} P_U \text{)}, \mathbb{M}_{p,\mathcal{P}} \right] \) from \( \psi \left[ \text{CR(} P_W \text{)} \right] \). Messages getseed were omitted for clarity.

value ♦ in case the simulator \( \nu \) never input dlv to the resource \( \text{CR(} P_W \text{)} \). From (6.3), we have

\[ d(R_{U|M_p Y Z}, T_{U|M_p Y Z}) \leq \varepsilon_3 \] \hspace{1cm} (6.7)

Note that in the random experiment defined by the joint distribution \( T_{W|M_p Y Z} \), messages with odd indices are actually sent by the (emulated) protocol \( \alpha \) while the output \( W \) of the resource \( \text{CR(} P_W \text{)} \) is fixed from the beginning of the random experiment and is unknown to \( \alpha \). Thus, the distribution \( T_{W|M_p Y Z} \) is such that \( W \) and a message \( M_j \) sent by \( \alpha \) are conditionally independent given the simulated seed \( U \) and the previous messages \( M^{j-1} \), i.e.,

\[ I^T(W; M_j | U M^{j-1}) = 0, \quad \text{for all odd } j \in \{1, \ldots, p\} \] \hspace{1cm} (6.8)

6.2.4 Combining the Simulatability Conditions

We will be interested into mixing the conditions on mutual information stated in (6.6) and (6.8). To do so, we first need to argue that the distributions \( S_{U|M_p W} \) and \( T_{U|M_p W} \) are statistically close. Let \( S((Y, Z) \neq (W, W)) \)
and $T ((Y, Z) \neq (W, W))$ denote the probability that the pair of random variables $(Y, Z)$ is different from $(W, W)$ in the random experiments defined by the distributions $S_{WUMpYZ}$ and $T_{WUMpYZ}$, respectively. We show that this probability must be small in both random experiments and since the same arguments apply to both cases we only detail the reasoning for $S ((Y, Z) \neq (W, W))$. Then, note that

$$
\text{d} (S_{YZ}, P_{WW}) \leq \text{d} (S_{YZ}, R_{YZ}) + \text{d} (R_{YZ}, P_{WW}) \leq \text{d} (S_{UMPYZ}, R_{UMPYZ}) + \text{d} (R_{YZ}, P_{WW}) \leq \varepsilon_2 + \varepsilon_1,
$$

where the last step follows from (6.1) and (6.2). Observe that by definition of the distribution $S_{WUMpYZ}$ the random variable $Y$ is either the error symbol $\bullet$ or $W$. This implies that a distinguisher trying to tell apart $\psi \text{CR} (P_W) \tau \beta$ from $\psi \text{CR} (P_W) \psi$ after having input getseed at both interfaces could have for advantage exactly $S ((Y, Z) \neq (W, W))$ by checking whether the random variable output at the left-interface is different from $\bullet$ and equals the random variable output at the right-interface. The maximum distinguishing advantage achieved by such a distinguisher is given by $\text{d} (S_{YZ}, P_{WW})$ and thus

$$
S ((Y, Z) \neq (W, W)) \leq \varepsilon_1 + \varepsilon_2,
\quad
T ((Y, Z) \neq (W, W)) \leq \varepsilon_1 + \varepsilon_3,
$$

(6.9)

where the second inequality follows by applying the same arguments to the distribution $T_{WUMpYZ}$.

Intuitively, since $(Y, Z)$ often equals $(W, W)$ the distributions $S_{UMPYZ}$ and $S_{UMPWW}$, and similarly for the distributions $T_{UMPYZ}$ and $T_{UMPWW}$, should be statistically close. More precisely, applying directly Lemma 2.3 shows that $\text{d} (S_{UMPWW}, S_{UMPYZ}) \leq \varepsilon_1 + \varepsilon_2$ and $\text{d} (T_{UMPYZ}, T_{UMPWW}) \leq \varepsilon_1 + \varepsilon_3$. Finally, combining via the triangle inequality these two previous upper bounds together with (6.5) and (6.7) gives

$$
\text{d} (S_{UMPW}, T_{UMPW}) \leq 2 (\varepsilon_1 + \varepsilon_2 + \varepsilon_3).
$$

(6.10)

### 6.3 Impossibility of CRA

We show in this section that the security constraints mentioned in Section 6.2 imply that the entropy gain $H(W) - H(U)$ of a $(P_U, P_W, p, \varepsilon)$-common randomness amplifier is close to 0 unless the distinguishing advantage quantified by $\varepsilon$ is large.
6.3 Impossibility of CRA

6.3.1 Perfect Case

For the sake of clarity, we first proceed to the perfect case ($\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0$) in Theorem 6.1 which implies as a trivial corollary that there exists no perfectly secure common randomness amplifier.

**Theorem 6.1.** Let $P_U$ and $P_W$ be two arbitrary distributions. Then, any protocol $\pi := (\alpha, \beta)$ and any simulator $\sigma := (\nu, \tau)$ are such that

$$\left[ \text{CRS} \left( P_U \right), \leftarrow \text{M}_p \right] \xrightarrow{(\pi, \sigma, 0)} \text{CR}_\psi \left( P_W \right) \implies H(W) \leq H(U) .$$

**Proof.** Assume that there exists a protocol $(\alpha, \beta)$ which perfectly constructs the common randomness resource $\text{CR}_\psi \left( P_W \right)$ from $\left[ \text{CRS} \left( P_U \right), \leftarrow \text{M}_p \right]$ and with respect to a simulator $(\nu, \tau)$, for some distributions $P_U$ and $P_W$.

Then, the availability condition implies via (6.4) that $R_{YZ} = P_{WW}$ and consequently $Y = Z$. Thus,

$$H(W) = H^R(Y) = I^R(Y;Z) .$$

The chain rule for mutual information implies that the last term can be upper bounded as follows

$$I^R(Y;Z) \leq I^R(Y;UM^PZ) = I^R(Y;U) + I^R(Y;M^P|U) + I^R(Y;Z|UM^P) \leq H^R(U) + I^R(Y;M^P|U) ,$$

where the last inequality follows from the fact that the output $Y$ of $\alpha$ and the output $Z$ of $\beta$ are conditionally independent given the seed $U$ and the messages $M^P$, i.e., $I^R(Y;Z|UM^P) = 0$.

To finish the proof it suffices therefore to show that $I^R(Y;M^P|U) = 0$. To see that, recall that (6.5) and (6.7) imply for $\varepsilon_2 = \varepsilon_3 = 0$ that

$$R_{UMPYZ} = S_{UMPYZ} = T_{UMPYZ}$$

and thus any information-theoretic measure involving these distributions are the same and in particular $I^R(Y;M^P|U) = I^S(Y;M^P|U)$.

Equation (6.9) and Lemma 2.3 imply then that $S_{UMPYZ} = S_{UMPWW}$ and $T_{UMPYZ} = T_{UMPWW}$. Thus,

$$I^S(Y;M^P|U) = I^S(W;M^P|U) = \sum_{j \in \{1, \ldots, p\}} I^S(W;M_j|UM^{j-1}) .$$
Finally, (6.6) implies $I^S(W; M_j | UM^{j-1}) = 0$ for a message with an even index; while $S_{UMPW} = T_{UMPW}$ and (6.8) imply

$$I^S(W; M_j | UM^{j-1}) = I^T(W; M_j | UM^{j-1}) = 0,$$

for a message with an odd index and overall $I^S(W; M^p | U) = 0$. 

### 6.3.2 Statistical Case

In this section we show the impossibility of statistically secure common randomness amplifiers. The proof of this impossibility result stated in Theorem 6.2 follows mostly that of Theorem 6.1, but requires in addition a technical tool to argue that if two distributions are statistically close, then so are any information-theoretic measures on these distributions.

#### From statistical distance to entropies.

This translation between statistical distance and information-theoretic quantities will be based on Fano’s inequality stated in Lemma 2.1 and on Lemma 2.4. As a consequence, the function $(d, k) \mapsto d \cdot \log_2 (k - 1) + h(d)$, where recall that $h(d)$ denotes the binary entropy function introduced in Section 2.2.1, will play a key role in the proof of Theorem 6.2 and we therefore start by stating some basic properties of this function.

**Lemma 6.1.** The function $F : [0, 1] \times \mathbb{N} \setminus \{0, 1\} \rightarrow \mathbb{R}$, where

$$F(d, k) := d \cdot \log_2 (k - 1) + h(d), \quad (6.11)$$

for all $d \in [0, 1]$ and $k \in \mathbb{N} \setminus \{0, 1\}$, is such that:

1. For any fixed $d \in [0, 1]$, the function $F(d, \cdot)$ is non-decreasing over $\mathbb{N} \setminus \{0, 1\}$.

2. For any fixed $k \in \mathbb{N} \setminus \{0, 1\}$, the function $F(\cdot, k)$ is non-decreasing over the interval $[0, 1 - \frac{1}{k}]$.

3. For any fixed $k \in \mathbb{N} \setminus \{0, 1\}$, the function $F(\cdot, k)$ is subadditive, i.e., $F(d_1 + d_2, k) \leq F(d_1, k) + F(d_2, k)$ for any $d_1, d_2 \in [0, 1]$ such that $d_1 + d_2 \leq 1$.

**Proof.** The first property trivially follows from the fact that the function $\log_2$ is non-decreasing over $\mathbb{N} \setminus \{0\}$. The second property follows by taking the derivative of $F(d, k)$ with respect to $d$, i.e., $F'(d, k) = \log_2 (k - 1) + \log_2 \left(\frac{1}{d} - 1\right) > 0$ for all $0 < d < 1 - \frac{1}{k}$. 

The last property basically follows from the fact that $F(0, k) = 0$ and
the function $F(\cdot, k)$ is concave, for all $k \in \mathbb{N} \setminus \{0, 1\}$. To see that, note that
$F''(d, k) = h''(d)$ and since $h(\cdot)$ is a well-known concave function then so is
$F(\cdot, k)$. Consider two reals $d_1, d_2 \in [0, 1]$ such that $d_1 + d_2 \leq 1$. If $d_1 + d_2 = 0$
then $d_1 = d_2 = 0$ and trivially $F(d_1 + d_2, k) \leq F(d_1, k) + F(d_2, k)$. Otherwise, if $d_1 + d_2 > 0$, then the concavity of the function $F(\cdot, k)$ implies that
\[
F(d_1, k) = F\left(\frac{d_2}{d_1 + d_2} \cdot 0 + \frac{d_1}{d_1 + d_2} (d_1 + d_2), k\right)
\geq \frac{d_2}{d_1 + d_2} \cdot F(0, k) + \frac{d_1}{d_1 + d_2} \cdot F(d_1 + d_2, k)
= \frac{d_1}{d_1 + d_2} \cdot F(d_1 + d_2, k),
\]
and it can similarly be proved that $F(d_2, k) \geq \frac{d_2}{d_1 + d_2} \cdot F(d_1 + d_2, k)$. Adding both lower bounds finishes proving that the function $F(\cdot, k)$ is
subadditive.

The following technical lemma shows that if we know an upper bound
on the statistical distance between two distributions, then we can also
upper bound the difference of any (entropy-based) information-theoretic
measure between these two distributions.

**Lemma 6.2.** Consider a $p$-tuple of random variables $X^p := (X_1, \ldots, X_p)$
over some set $\Omega$ distributed in one random experiment according to $P_{X^p}$
and in another random experiment according to $Q_{X^p}$. Then, for any subsets
$S_1, S_2, S_3 \subseteq \{X_1, \ldots, X_p\}$ of random variables, any $k$ and $d$ such that
$|\Omega| \leq k$ and $d(P_{X^p}, Q_{X^p}) \leq d \leq 1 - \frac{1}{k}$,
\[
|H^P (S_1) - H^Q (S_1)| \leq d \cdot \log_2 (k - 1) + h(d),
|H^P (S_1 | S_2) - H^Q (S_1 | S_2)| \leq 2 (d \cdot \log_2 (k - 1) + h(d)),
|I^P (S_1 ; S_2) - I^Q (S_1 ; S_2)| \leq 3 (d \cdot \log_2 (k - 1) + h(d)),
|I^P (S_1 ; S_2 | S_3) - I^Q (S_1 ; S_2 | S_3)| \leq 4 (d \cdot \log_2 (k - 1) + h(d)).
\]

**Proof.** We start by proving the first inequality since the others will follow
from it. Let $S_1 \subseteq \{X_1, \ldots, X_p\}$ be a subset of random variables over some
set $S_1$. Note that $|S_1| \leq |\Omega| \leq k$ and $d(P_{S_1}, Q_{S_1}) \leq d (P_{X^p}, Q_{X^p}) \leq d \leq
$1 - \frac{1}{k}$. Then, Lemma 2.4 implies that,

$$|H^P(S_1) - H^Q(S_1)| \leq d(P_{S_1}, Q_{S_1}) \cdot \log_2(|S_1| - 1) + h(d(P_{S_1}, Q_{S_1}))$$

$$\leq d(P_{S_1}, Q_{S_1}) \cdot \log_2(k - 1) + h(d(P_{S_1}, Q_{S_1}))$$

$$\leq d \cdot \log_2(k - 1) + h(d),$$

where the last two steps follow from the monotonicity of the function $F$ stated in Lemma 6.1.

The other inequalities mainly follow from the triangle inequality. More precisely, let $S_2, S_3 \subseteq \{X_1, \ldots, X_p\}$ be two sets of random variables. Recall that $H^P(S_1 | S_2) = H^P(S_1 S_2) - H^P(S_2)$ where notice that $H^P(S_1 S_2) = H^P(S_1 \cup S_2)$, and similarly for $H^Q(S_1 | S_2)$. Then, the first inequality proved in the previous paragraph implies

$$|H^P(S_1 | S_2) - H^Q(S_1 | S_2)| \leq |H^P(S_1 S_2) - H^Q(S_1 S_2)|$$

$$+ |H^P(S_2) - H^Q(S_2)|$$

$$\leq 2(d \cdot \log_2(k - 1) + h(d)).$$

The remaining inequalities are proved in the same manner and so we only sketch their proof. Recall that $I^P(S_1 ; S_2) = H^P(S_1) - H^P(S_1 | S_2)$ and similarly for $I^Q(S_1 ; S_2)$. Thus, the last two inequalities proved above imply

$$|I^P(S_1 ; S_2) - I^Q(S_1 ; S_2)| \leq |H^P(S_1) - H^Q(S_1)|$$

$$+ |H^P(S_1 | S_2) - H^Q(S_1 | S_2)|$$

$$\leq 3(d \cdot \log_2(k - 1) + h(d)).$$

Finally, notice that $I^P(S_1 ; S_2 | S_3) = H^P(S_1 | S_3) - H^P(S_1 | S_2 S_3)$ and similarly for $I^Q(S_1 ; S_2 | S_3)$. Thus,

$$|I^P(S_1 ; S_2 | S_3) - I^Q(S_1 ; S_2 | S_3)|$$

$$\leq |H^P(S_1 | S_3) - H^Q(S_1 | S_3)| + |H^P(S_1 | S_2 S_3) - H^Q(S_1 | S_2 S_3)|$$

$$\leq 4(d \cdot \log_2(k - 1) + h(d)),$$

where the last step follows from the upper bound on the difference of conditional entropies proved above.

The next theorem implies that any $(P_U, P_W, p, \varepsilon)$-common randomness amplifier is either trivially insecure, i.e., $\varepsilon_1 + \varepsilon_2 + \varepsilon_3$ is (almost) at least $\frac{1}{2}$, or the entropy gain $H(W) - H(U)$ is “small”, i.e., upper bounded by a
quantity which goes to zero as $\varepsilon_1 + \varepsilon_2 + \varepsilon_3$ goes to zero. In particular, if the common randomness amplifier considered is both statistically secure and efficient, then its entropy gain is negligible.

**Theorem 6.2.** Let $P_U$ and $P_W$ be two arbitrary distributions over $\mathcal{U}$ and $\mathcal{W}$, respectively. Consider a triple $\varepsilon := (\varepsilon_1, \varepsilon_2, \varepsilon_3)$ of non-negative reals and a protocol $\pi := (\alpha, \beta)$ together with a simulator $\sigma := (\nu, \tau)$ such that

$$\left[\text{CRS} \left( P_U \right), \mathcal{M}_p \right] \xrightarrow{TDP} \text{CR}_\psi \left( P_W \right).$$

If $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 \leq \frac{1}{2} \left( 1 - \frac{1}{|\Omega|} \right)$ with $\Omega := U \times M^p \times W$, then

$$H(W) \leq H(R(Y)) + (4p + 7) \cdot F(\varepsilon_1 + \varepsilon_2 + \varepsilon_3, |\Omega|),$$

where the function $F$ was defined in (6.11).

Before proving Theorem 6.2 let us comment on the upper bound obtained. First, note that if the protocol $\pi$ is efficient communication-wise, in the sense that the total number of bits input and output by both $\alpha$ and $\beta$ is at most a polynomial of some security parameter $\lambda$, then $(4p + 7) \cdot F(\varepsilon_1 + \varepsilon_2 + \varepsilon_3, |\Omega|)$ will be a negligible function of $\lambda$ if all $\varepsilon_1, \varepsilon_2$ and $\varepsilon_3$ are also negligible, since $p$ and $\log_2 (|\Omega| - 1)$ are in this case a polynomial function of $\lambda$. Second, the bound obtained in Theorem 6.2 could be made tighter, at the cost of becoming more complex and less manageable, if instead of considering the sum $\varepsilon_1 + \varepsilon_2 + \varepsilon_3$ one considers separately each $\varepsilon_j$ as will be shown in the proof.

**Proof (of Theorem 6.2).** Assume that there exists a protocol $\pi := (\alpha, \beta)$ which constructs within $\varepsilon = (\varepsilon_1, \varepsilon_2, \varepsilon_3)$ the (filtered) common randomness resource $\text{CR}_\psi (P_W)$ from the assumed resource $\left[\text{CRS} \left( P_U \right), \mathcal{M}_p \right]$ and with respect to a simulator $\sigma := (\nu, \tau)$, for some distributions $P_U$ and $P_W$ and where $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 \leq \frac{1}{2} \left( 1 - \frac{1}{|\Omega|} \right)$. Throughout the proof, we will use the function $F(d, k) := d \cdot \log_2 (k - 1) + h(d)$ introduced in (6.11).

**Step 1.** First, note that the distance constraint stemming from the availability condition and stated in (6.4) shows that $d(R_Y, P_W) \leq \varepsilon_1 \leq 1 - \frac{1}{|\Omega|}$. Second, since $|W| \leq |\Omega|$, we can apply Lemma 6.2 to obtain

$$H(W) \leq H(R(Y)) + F(\varepsilon_1, |\Omega|).$$

Equation (6.4) also implies that the probability that $Y$ is different from $Z$ in the real random experiment defined by the distribution $R_{U, M^p, Y, Z}$.
denoted \( R(Y \neq Z) \), is at most \( \varepsilon_1 \). Then, Fano’s inequality in Lemma 2.1 and the monotonicity of the function \( F \) shown in Lemma 6.1 give

\[
H^R(Y) = I^R(Y; Z) + H^R(Y | Z) \\
\leq I^R(Y; Z) + F(R(Y \neq Z), |W|) \\
\leq I^R(Y; Z) + F(\varepsilon_1, |\Omega|)
\]

Finally, the mutual information term can be upper bounded exactly as in the proof of Theorem 6.1 to obtain \( I^R(Y; Z) \leq H^R(U) + I^R(Y; M^p | U) \). Overall, we therefore have the following upper bound

\[
H(W) \leq H^R(U) + I^R(Y; M^p | U) + 2 \cdot F(\varepsilon_1, |\Omega|).
\]

**Step 2.** The next step of the proof consists of showing that the mutual information term \( I^R(Y; M^p | U) \) is “small”. We will proceed in two steps, the first one being to show that this mutual information term is close to \( IS(Y; M^p | U) \). The simulatability condition against a malicious Bob implies via (6.5) that \( d(R_{U_MY}, S_{U_MY}) \leq \varepsilon_2 \leq 1 - \frac{1}{|\Omega|} \) and Lemma 6.2 then implies that

\[
I^R(Y; M^p | U) \leq IS(Y; M^p | U) + 4 \cdot F(\varepsilon_2, |\Omega|).
\]

The probability that \( Y \) is different from \( W \) is at most \( \varepsilon_1 + \varepsilon_2 \) by (6.9). Fano’s inequality in Lemma 2.1 and the monotonicity of \( F \) in Lemma 6.1 then imply that \( H^S(Y | W) \leq F(\varepsilon_1 + \varepsilon_2, |\Omega|) \). Thus, we can replace \( Y \) by \( W \) in the previous mutual information term as follows

\[
I^S(Y; M^p | U) \leq I^S(W; M^p | U).
\]

where the third inequality follows from the fact that conditioning reduces entropy, i.e., \( I^S(Y; M^p | U) \leq H^S(Y | U) \leq H^S(Y | W) \). Overall, combining the equations in this step gives

\[
I^R(Y; M^p | U) \leq I^S(W; M^p | U) + F(\varepsilon_1 + \varepsilon_2, |\Omega|) + 4 \cdot F(\varepsilon_2, |\Omega|).
\]

**Step 3.** Recall that \( I^S(W; M^p | U) = \sum_{j \in \{1, \ldots, p\}} I^S(W; M_j | U M^{j-1}) \) and (6.6) implies that all the terms where \( j \) is even are null. To show
that the rest of the terms must also be “small”, let us consider the mutual information term $I^S(W; M_j | UM^{j-1})$ for an odd $j$. Note that (6.10) implies that $d(S_{UM^jW}, T_{UM^jW}) \leq 2(\varepsilon_1 + \varepsilon_2 + \varepsilon_3) \leq 1 - \frac{1}{|\Omega|}$ and of course $|U \times M^j \times W| \leq |\Omega|$. Then, Lemma 6.2 and the fact that $I^T(W; M_j | UM^{j-1}) = 0$ by (6.8) imply that

$$I^S(W; M_j | UM^{j-1}) \leq 4 \cdot F(2(\varepsilon_1 + \varepsilon_2 + \varepsilon_3), |\Omega|),$$

for all odd $j \in \{1, \ldots, p\}$. Thus, summing over all $\frac{p}{2}$ odd integers in $\{1, \ldots, p\}$ gives

$$I^S(W; M^p | U) \leq 2p \cdot F(2(\varepsilon_1 + \varepsilon_2 + \varepsilon_3), |\Omega|) \leq 4p \cdot F(\varepsilon_1 + \varepsilon_2 + \varepsilon_3, |\Omega|),$$

where the last inequality follows from the subadditivity of the function $F$ shown in Lemma 6.1.

The proof is basically finished by combining all the previous equations. To make the final bound more manageable, any approximation term involving the function $F(\cdot, |\Omega|)$ will be upper bounded by $F(\varepsilon, |\Omega|)$ where $\varepsilon := \varepsilon_1 + \varepsilon_2 + \varepsilon_3$, which is the biggest of such terms encountered so far. Overall, we thus have

$$H(W) \leq H^R(U) + I^R(Y; M^p | U) + 2 \cdot F(\varepsilon, |\Omega|) \leq H^R(U) + I^S(W; M^p | U) + 7 \cdot F(\varepsilon, |\Omega|) \leq H^R(U) + (4p + 7) \cdot F(\varepsilon, |\Omega|),$$

which finishes the proof. \qed
Chapter 7

Unfair Coin Tossing

In this chapter, we show that Blum’s coin tossing protocol [Blu81] constructs an unfair coin tossing resource, where a party can bias the bit output to the other party. Blum’s protocol is formalized as a pair of converters in Section 7.1 and we introduce the unfair coin tossing resource in Section 7.2. Finally, we show the actual construction in Section 7.3.

This chapter is stated in the setting of two distrustful parties described in Section 2.5.3. Resources in this chapter have therefore a left- and a right-interface, and the set \( \mathcal{I} \) of interfaces is \( \mathcal{I} := \{ \text{left}, \text{right} \} \).

7.1 Blum’s Coin Tossing Protocol

**Assumed resources.** The construction statement shown below assumes the availability of a bit commitment resource (with abort) \( \text{COM} \) and of a communication channel denoted \( \leftarrow \). The communication channel \( \leftarrow \) simply forwards any bit input by Bob (at the right-interface) to Alice (at the left-interface).

Informally, the resource \( \text{COM} \) allows Alice to commit to a single bit \( x \) and to later choose when this bit is to be output to Bob. Alternatively, this second phase can be aborted by Alice, in which case the value \( x \) which was initially committed is never revealed and Bob is instead informed that Alice aborted. In practice, aborting the commitment would correspond to a situation where Bob waited for the opening of a committed value beyond a certain threshold. The commitment resource \( \text{COM} \) is described more formally in System 7.1.
System 7.1: Bit commitment resource COM with abort

\[ m := \diamond \text{ and } dlv_{\text{right}} := 0 \]

on first input (commit, \( x \in \{0, 1\} \)) at left
\[
\begin{align*}
& m := x \\
& \text{output committed at right}
\end{align*}
\]

on input open at left
\[
\begin{align*}
& \text{if } m \neq \diamond \text{ and } dlv_{\text{right}} = 0 \text{ then} \\
& \quad dlv_{\text{right}} := 1 \\
& \quad \text{output } m \text{ at right}
\end{align*}
\]

on input abort at left
\[
\begin{align*}
& \text{if } m \neq \diamond \text{ and } dlv_{\text{right}} = 0 \text{ then} \\
& \quad dlv_{\text{right}} := 1 \\
& \quad \text{output abort at right}
\end{align*}
\]

Blum’s protocol. We model Blum’s coin tossing protocol as a pair of converters \( \text{toss} = (\text{bet}, \text{flip}) \), where \( \text{bet} \) is Alice’s converter and \( \text{flip} \) is Bob’s. Alice initiates the coin toss by committing to a random bit \( x \) via the resource \( \text{COM} \). Afterwards, Bob sends to Alice via the communication channel \( \leftarrow \) his own random bit \( y \). Finally, Alice opens her commitment after having received \( y \) and outputs a coin with value \( c := x \oplus y \). Upon receiving the committed value \( x \), Bob also outputs a coin \( c' := x \oplus y \), but if the commitment was instead aborted (by a malicious Alice), Bob outputs an independent random bit \( c' \). Since the commitment resource \( \text{COM} \) let Bob know when Alice aborted, the protocol also divulges this information, i.e., together with the coin \( c' \) Bob outputs a bit indicating whether or not Alice tried to influence his output by aborting prematurely. The corresponding converters \( \text{bet} \) and \( \text{flip} \) are described more formally in Systems 7.2 and 7.3 and a honest execution of Blum’s protocol is shown in Figure 7.1.

Note that even if Bob were able to abort the protocol, he would not be able to bias Alice’s output when she is the initiator of the coin toss. For this reason the channel \( \leftarrow \), in contrast to the commitment resource \( \text{COM} \), does not have a special input \text{abort} to model abort from a (malicious) Bob.
7.2 Necessity of an Unfair Coin Tossing Resource

**System 7.2**: Converter bet

\[ x := \diamond \quad \text{and} \quad c := \diamond \]

**on first input** init at out

- \( x \leftarrow \{0, 1\} \)

**output** (commit, \( x \)) at in₁

**on input** \( y \in \{0, 1\} \) at in₂

- if \( x \neq \diamond \) and \( c = \diamond \) then
  - \( c := x \oplus y \)
  - **output** open at in₁
  - **output** \( c \) at out

**System 7.3**: Converter flip

\[ \text{init}_{\text{left}} := 0 \quad \text{and} \quad \text{dlv}_{\text{left}} := 0 \]

\[ y := \diamond \quad \text{and} \quad c' := \diamond \]

**on first input** committed at in₁

- \( \text{init}_{\text{left}} := 1 \)
- **output** init at out

**on input** toss at out

- if \( \text{init}_{\text{left}} = 1 \) and \( \text{dlv}_{\text{left}} = 0 \) then
  - if \( y = \diamond \) then \( y \leftarrow \{0, 1\} \)
  - \( \text{dlv}_{\text{left}} := 1 \)
- **output** \( y \) at in₂

**on input** \( x \in \{0, 1\} \) at in₁

- if \( c' = \diamond \) then
  - if \( y = \diamond \) then \( y \leftarrow \{0, 1\} \)
  - \( c' := x \oplus y \)
  - **output** \((0, c')\) at out

**on input** abort at in₁

- if \( c' = \diamond \) then
  - \( c' \leftarrow \{0, 1\} \)
  - **output** \((1, c')\) at out

7.2 Necessity of an Unfair Coin Tossing Resource

**Desired resource.** The goal of the protocol \( \text{toss} = (\text{bet, flip}) \) is to construct a coin tossing resource outputting the same uniform random bit to both parties. However, it is not hard to see that a malicious Alice can influence Bob’s output by aborting the commitment phase. To see that, notice that when Alice is supposed to open her commitment she already knows the what the protocol’s outcome would normally be \( x \oplus y \), since she knows the bit \( x \) she initially committed to as well as the bit \( y \) sent by Bob. A malicious Alice can therefore influence Bob’s output, say towards 0, as follows. If \( x \oplus y = 0 \), then Alice opens her commitment as she would normally do and Bob outputs 0. Otherwise, if \( x \oplus y = 1 \), then Alice aborts her commitment and Bob outputs a uniform random bit. Overall, this malicious strategy forces Bob to output 0 with probability \( \frac{3}{4} \), which corresponds to a bias towards 0 of \( \frac{1}{2} \).
Figure 7.1. Blum’s coin tossing protocol. The assumed resource, a commitment resource COM and a channel $\leftarrow$, with protocol converters bet and flip attached to interfaces left and right, denoted bet $[COM, \leftarrow]$ flip. The coin toss is initialized by the converter bet committing to a random bit $x$. The converter flip decides when the coin is tossed by sending a random bit $y$ to bet, which only then opens its commitment. Both converters output a coin with value $x \oplus y$.

Thus, the coin tossing resource constructed by the protocol toss needs to be unfair in the sense that Alice should be able to see the value of the coin toss before Bob, and depending on this value she could also try to bias the bit output to Bob. This unfair coin tossing resource $UCT$ is parameterized by a value $a \in [0, 1]$ and the coin $C'$ outputs to Bob by $UCT^a$ is guaranteed to have an absolute bias of at most $a$, i.e., $|b(C')| \leq a$ where recall that $b(C') = 1 - 2 \cdot P_{C'}(1)$. Despite this unavoidable biased output, Bob does have two additional advantages in using the resource $UCT^a$. First, $UCT^a$ let Bob decide when Alice shall receive her coin toss, the reason being that in Blum’s protocol toss Alice only knows the protocol’s output $x \oplus y$ once Bob decided to send his own random bit $y$. Second, $UCT^a$ outputs a second bit to Bob stating whether Alice tried to bias his output, similarly to how the resource COM informs Bob when Alice aborted her commitment. The unfair coin tossing resource $UCT^a$ is described more formally in System 7.4.

Filtered resources. The ability to influence Bob’s output is only present if Alice behaves dishonestly. We will therefore consider a filter $\Psi := (\Psi, 1)$, where the converter $\Psi$ acts as follows: Upon receiving a bit $c$ at its interface in, the converter $\Psi$ outputs $(bias, 0)$ at its interface in and outputs $c$ at its interface out; while any other input, at its in- or out-interface, is directly forwarded to the other interface. The corresponding filtered resource $UCT_{\Psi}^a$ guarantees Alice to have access to $\Psi UCT^a$, a resource outputting a perfect
7.3 Constructing UCT

The next theorem shows that Blum’s protocol perfectly constructs a filtered $\frac{1}{2}$-biased unfair coin tossing resource $\text{UCT}^{a}_{\Psi}$. As explained in the paragraph above, it is always possible for a malicious Alice to give a $\frac{1}{2}$-bias to Bob’s output (towards 0 or 1), meaning that Blum’s protocol cannot construct a less biased coin tossing resource $\text{UCT}^{a}_{\Psi}$ where $a < \frac{1}{2}$.

**Theorem 7.1.** Blum’s coin tossing protocol $\text{toss} = (\text{bet}, \text{flip})$ described in Systems 7.2 and 7.3 is such that

$$[	ext{COM, }] \xrightarrow{\text{toss}, \sigma_{\text{toss}}, 0}_{\text{TDP}} \text{UCT}^{\frac{1}{2}}_{\Psi},$$

where the simulator $\sigma_{\text{toss}} := (\sigma_{\text{bet}}, \sigma_{\text{flip}})$ is described below in Systems 7.5 and 7.6.
Proof. Before describing the simulator \((\sigma_{\text{bet}}, \sigma_{\text{flip}})\), let us recall that according to Definition 2.7 we need to show the following three conditions:

\[
\Delta^D(\text{bet} [\text{COM}, \leftarrow \rightarrow] \text{flip}, \Psi \text{ UCT}^{\frac{1}{2}}) = 0, \quad (7.1)
\]
\[
\Delta^D(\text{bet} [\text{COM}, \leftarrow \leftarrow], \Psi \text{ UCT}^{\frac{1}{2}} \sigma_{\text{flip}}) = 0, \quad (7.2)
\]
\[
\Delta^D(\left[\text{COM}, \leftarrow \rightarrow\right] \text{flip}, \sigma_{\text{bet}} \text{ UCT}^{\frac{1}{2}}) = 0, \quad (7.3)
\]

for all distinguishers \(D\).

The availability condition (7.1) trivially holds since in the real world both parties output a coin with value \(x \oplus y\), where \(x\) and \(y\) are two uniform and independently chosen bits.

The simulatability condition (7.2) against a malicious Bob is also readily verified by considering the following simulation strategy \(\sigma_{\text{flip}}\) detailed in System 7.5. Upon receiving the initialization message init from the unfair coin tossing resource \(\text{UCT}^{\frac{1}{2}}\) at its interface in, the simulator \(\sigma_{\text{flip}}\) forwards the message committed at its interface out\(_1\) to mimic the behavior of the right-interface of the commitment resource \(\text{COM}\) in the real world. When a bit \(y\) is input to its out\(_2\)-interface, corresponding in the real world to a message input to the channel \(\leftarrow\), the simulator \(\sigma_{\text{flip}}\) sends the coin \(c\) tossed by \(\text{UCT}^{\frac{1}{2}}\) to Alice (through the filter \(\Psi\)) by outputting \(\text{toss}\) at its in-interface. Finally, when the simulator \(\sigma_{\text{flip}}\) receives the coin \(c\), it emulates the opening of the commitment by outputting \(c \oplus y\) in order to be consistent with the value that would be output in the real world.

**System 7.5: Simulator \(\sigma_{\text{flip}}\)**

\[
\begin{align*}
\text{init}_{\text{left}} &:= 0 \text{ and } m := \diamond \\
\text{on first input } \text{init at in} &\quad \text{output committed at out}_1 \\
\text{on input } y \in \{0, 1\} \text{ at out}_2 &\quad \text{if } \text{init}_{\text{left}} = 1 \text{ and } m = \diamond \text{ then} \\
&\quad \quad m := y \\
&\quad \quad \text{output toss at in} \\
\text{on input } (0, c) \text{ at in} &\quad \text{output } c \oplus m \text{ at out}_1
\end{align*}
\]

**System 7.6: Simulator \(\sigma_{\text{bet}}\)**

\[
\begin{align*}
m &:= \diamond \\
\text{on first input } \text{(commit, x) at out}_1 &\quad m := x \\
\text{on input open at out}_1 &\quad \text{if } m \neq \diamond \text{ then output (bias, 0) at in} \\
\text{on input abort at out}_1 &\quad \text{if } m \neq \diamond \text{ then output (bias, 1) at in} \\
\text{on input c at in} &\quad \text{output } c \oplus m \text{ at out}_2
\end{align*}
\]
In order to prove the simulatability condition (7.3) against a malicious Alice, consider the following simulation strategy $\sigma_{\text{bet}}$ described in System 7.6. When the simulator $\sigma_{\text{bet}}$ receives the message $(\text{commit}, x)$, corresponding in the real world to the bit $x$ to be committed, $\sigma_{\text{bet}}$ initializes the coin tossing resource $\text{UCT}^{\frac{1}{2}}$ by outputting init at its interface in. Upon receiving the actual coin toss $c$ from the resource, the simulator $\sigma_{\text{bet}}$ emulates the transmission of a message from Bob by outputting $c \oplus x$. Finally, when the simulator receives the opening message open it outputs $(\text{bias}, 0)$ to the resource $\text{UCT}^{\frac{1}{2}}$ in order for Bob to retrieve the same coin toss $c' = c$. Note that so far the simulation is perfect. If instead the simulator $\sigma_{\text{bet}}$ received the abort message abort, then it outputs $(\text{bias}, 1)$ to $\text{UCT}^{\frac{1}{2}}$. In this case, the bit $c'$ output to Bob is defined as $c' := c \oplus n$, where $n$ is a sample of a uniform and independent binary random variable so that $c'$ is also uniformly and independently distributed, therefore also perfectly emulating what happens in the real world. 

$\square$
Bibliography


(Cited on page 8.)

(Cited on page 46.)

(Cited on pages 41 and 44.)

(Cited on page 15.)

(Cited on pages 16 and 143.)

(Cited on page 8.)

(Cited on page 17.)

(Cited on page 8.)

(Cited on pages 12 and 44.)

(Cited on page 44.)

(Cited on page 44.)

(Cited on page 33.)
(Cited on pages 4, 6, 7, 8, 11, 49, 79, and 84.)

(Cited on page 15.)

(Cited on page 8.)

(Cited on page 15.)

(Cited on page 8.)

(Cited on pages 41, 44, and 115.)


(Cited on page 85.)

(Cited on page 104.)

(Cited on page 3.)

(Cited on pages 3, 8, and 104.)

(Cited on page 3.)

(Cited on page 3.)


(Cited on pages 13, 25, 31, 33, 34, and 35.)

(Cited on pages 2, 19, 25, 29, and 30.)

(Cited on pages 34 and 35.)

(Cited on page 8.)

(Cited on page 17.)

(Cited on pages 32 and 33.)

(Cited on pages 2, 10, 19, 25, 27, 28, and 29.)


(Cited on page 8.)

(Cited on page 8.)

(Cited on page 10.)

(Cited on page 4.)

(Cited on page 11.)

(Cited on page 24.)

(Cited on page 44.)

(Cited on page 1.)

(Cited on page 9.)
(Cited on page 9.)

(Cited on pages 15 and 16.)

(Cited on page 13.)

(Cited on page 10.)

(Cited on page 1.)