On Cryptographic Primitives Based on Noisy Channels

Kirill Morozov

PhD Dissertation

Department of Computer Science
University of Aarhus
Denmark

On Cryptographic Primitives Based on Noisy Channels

A Dissertation
Presented to the Faculty of Science
of the University of Aarhus
in Partial Fulfilment of the Requirements for the
PhD Degree

by
Kirill Morozov
March 26, 2005

Abstract

The primitives of Oblivious Transfer (OT) and Bit Commitment (BC) are fundamental in the cryptographic protocol design. OT is a complete primitive for secure two-party computation, while zero-knowledge proofs are based on BC. In this work, the implementations of OT and BC with unconditional security for both parties are considered. The security of these protocols does not depend on unproven intractability assumptions. Instead, we assume that the players are connected by noisy channels. This is a very natural assumption since noise is inherently present in the real communication channels.

We present and prove secure a protocol for OT based on a Discrete Memoryless Channel (DMC) with probability transition matrix of a general form. The protocol is secure for any non-trivial DMC. Some generalisations to this protocol for the particular case of Binary Symmetric Channel (BSC) are presented and their asymptotic behaviour is analysed.

The security of OT and BC based on BSC is also analysed in the non-asymptotic case. We derive relations for the failure probabilities depending on the number of channel uses establishing trade-offs between their communication complexity on the one hand and the security on the other hand.

We consider a modification to the Universally Composable (UC) framework for the case of unconditional two-party protocols. We argue that this modification is valid hereby preparing a ground for our results concerning OT based on Unfair Noisy Channels (UNC).

In contrast to the noise models mentioned above, a corrupted party is given a partial control over the randomness introduced by UNC. We introduce a generic "compiler" which transforms any protocol implementing OT from a passive version of UNC and secure against passive cheating into a protocol that uses UNC for communications and builds an OT secure against active cheating. We exploit this result and a new technique for transforming between the weaker versions of OT in order to obtain new possibility results for OT based on UNC.

Acknowledgements

I am deeply grateful to Ivan Damgård for his invaluable help and encouragement throughout my PhD studies. He always had time (and enormous patience) for discussions with me.

Special thanks go to Valeri Korjik for having been my supervisor during my studies in Saint-Petersburg. He first introduced me to cryptography and it was always a pleasure to work with him. Many thanks to Louis Salvail for his support, encouragement and discussions (not only scientific). Special thanks go to Claude Crépeau and Stefan Wolf for the beautiful time which I was having when visiting McGill University.

I am very grateful to the academic and technical stuff at the University of Aarhus for always being helpful, on-time and up-to-date, in particular (but not exclusively): Mogens Nielsen, Uffe Engberg, Janne Christensen, Karen Møller, Hanne Jensen, Helle Nielsen, Ingrid Larsen, Sanne Jensen, DAIMI computer support team, DAIMI library.

I am very thankful to Jesús Almansa and Christian Schaffner for proofreading and many helpful comments, suggestions and corrections to this work.

I would like to thank my other co-authors, colleagues and friends, Jörg Abendroth, Saurabh Agarwal, Abdul Ahsan, Geneviève Arboit, Pablo Arrighi, Claus Brabrand, Marco Carbone, Aske Christensen, Martin Courcheesne, Stefan Dantchev, Carlton Davis, Simon-Pierre Desrosiers, Kasper Dupont, Gregory Estren, Serge Fehr, Mayer Goldberg, Bernd Grobauer, Jens Groth, Nelly Fazio, Matthias Fitzi, Rico Jacob, Mads Jurik, Dan Hernest, Bartek Klin, Margarita Korovina, Maciej Koprowski, Sunil Kothari, Andrey Krasov, Dmitri Kushnir, Branimir Lambov, Jooyong Lee, Laurentiu Leustean, Jan Midtgaard, Giuseppe Milicia, Gabriel Moruz, Oliver Möller, Antonio Nicolosi, Jesper Nielsen, Thomas Pedersen, Mathieu Petitpas, Raymond Putra, George Savvides, Paweł Sobocinski, Martijn Stam, Hugo Touchette, Frank Valencia, Daniele Varacca, Anton Vershovski, the Fellowship of 3704 Peel, Skjoldhøj – Opgang 103 for interesting, nice and funny time.

I express my deep gratitude to the financial supporters of my PhD studies: the Danish National Research Foundation, the Danish Natural Sciences Research Council and eventually the millions of Danish taxpayers.

I am deeply indebted to my parents, Tatiana Morozova and Gennady Morozov for supporting me in all my undertakings and making my education possible at all. A very special thanks go to my family for encouraging me during my studies.

Kirill Morozov,
Århus, March 26, 2005.

Contents

Chapter 1

Introduction

For centuries since the invention of writing, *cryptography* was an art of exchanging messages privately between two or more parties who shared a common secret piece of information. The seminal work of Shannon [Sha49] turned cryptography into science but still the cryptographic systems (or ciphers) were used to encrypt and decrypt information based on a *secret key* shared by the parties.

The revolutionary idea of Diffie and Hellman [DH76] was to separate keys for encryption and decryption so that anyone can encrypt messages based on public information but only the designated receiver can decrypt them. This work gave an additional impulse to research in the area of *cryptographic protocols* where not a protection from the third party but a *mutual privacy* of the data used by the parties was concerned. We give an intuitive definition of *a protocol* as a "way for a number of parties to blend some secret information revealing part of it to the participants while keeping part of it secret" [Cré90]. The protocols were designed to perform the vast variety of tasks from secret-key exchange [DH76] and zero-knowledge proofs [GMR85] to electronic voting [CF85] and digital cash systems [CFN88], their participants from being just the parties had turned into the *players*.

At some point later, the results of more general nature were obtained. Yao [Yao82] presented a protocol for general problem of two-party computation for the two players who want to calculate a function on their inputs still keeping them private. Goldreich, Micali and Wigderson [GMW87] presented a way to obtain a secure multi-party protocol from its abstract specification. These works as well as the related results on multi-party computation by Chaum and Damgård with van de Graaf [CDG87] and with Crépeau [CCD88] pointed at the fact that there exist elementary protocols (also called *primitives*) which are essential for achieving more advanced ones, in particular, multi-party computation.

Those primitives of *Oblivious Transfer* (OT) and *Bit Commitment* (BC) appeared to be fundamental in cryptographic protocol design. An OT is a means to transmit data such that the sender is guaranteed that the data will be partially lost during the transmission, but he does not know what exactly the receiver gets. A BC scheme is a tool for transmitting an evidence about a piece of information without revealing the information itself. In our work, we focus on the case of *two-party* protocols where two players, a *sender* and

a *receiver* can be unambiguously identified, moreover the players are *mutually distrusting*, so the *dishonest* (also called *cheating* or *corrupted*) party plays a role of an *adversary*. We shall study OT and BC which are *unconditionally secure* for both sender and receiver and which are based on noisy channels.

1.1 Unconditional Security

There are essentially three "flavours" of security considered in cryptography, reflecting the computational power of the adversary and the probability that the security fails:

- *Computational security:* Security against an adversary which is assumed to be computationally bounded, in the sense that he can be modelled by a probabilistic polynomial time (PPT) Turing machine.
- *Perfect security:* Security against a computationally unbounded adversary, and no failure probability is allowed.
- *Unconditional (or information-theoretic) security:* Security against a computationally unbounded adversary which can compute everything that is uniquely defined from his point of view and a negligibly small failure probability is allowed.

The term "unconditional" is due to historical reasons meaning independency of computational power of the adversary. Nevertheless, after removing the assumption that adversary's computational ability is bounded, one has to make another assumption on players' capabilities and/or communications, otherwise it is well-known to be impossible to construct any primitive with unconditional security for *both* parties based on *noiseless* (or *error-free*) communication between them. Informally speaking, this assumption ensures that even a computationally unbounded cheater cannot recover some data involved in the player's interaction, so that information-theoretic methods can be used in the security proofs.

Our motivation in the study of unconditional cryptographic protocols comes from the fact that these protocols provide an information protection which is indeed secure "once forever", in spite of the growing power of modern computing systems, possible break-throughs in the complexity theory or technological achievements such as quantum computers. At the same time, the unconditional model rather than the perfect security model allows us to accommodate such an assumption as noisy communication between the parties since security failures are always possible in this case. On the other hand, once the properties of noise are (at least partially) known to the players, it may be possible to make sure that the failures almost never happen on average.

In this work, we focus on unconditional two-party primitives based on noisy communication between the parties. The noise is inherently present in any physical communication channel and, as a matter of fact, it is usually given "for free". Therefore, the study of the primitives based on noisy channels is

a problem which arises naturally and, at the same time, it is attractive from both theoretical and practical points of view. For the theory, it has always been important to figure out the weakest possible assumption which is sufficient to achieve those primitives and, consequently, secure multi-party computation. From the constructive side, despite of the fact that the general techniques are rarely used in practice, one can exploit the methods studied here when implementing practical protocols where one usually exchanges generality for efficiency.

1.2 Oblivious Transfer

The Oblivious Transfer primitive was independently introduced by Wiesner approximately in 1970 [Wie83] and by Rabin in 1981 [Rab81]. It was shown by Goldreich, Micali and Wigderson [GMW86] in the computational model and Kilian [Kil88] in the abstract model that OT suffices to perform secure two-party computations, presented by Yao in 1982 [Yao82], where the two players want to compute a function on their inputs in such a way that the inputs remain private and a correctness of the result is guaranteed.

One-out-of-two Oblivious Transfer[1] originates with [Wie83] being unpublished since approximately 1970. The construction was based on quantum mechanics principles and appeared under the label of "multiplexing". According to this primitive, a sender A with two bits b_0 and b_1 as input can interact with a receiver B with his input bit c also called the *selection bit*. A should learn nothing new from the protocol, whereas B should learn the *choice bit* b_c and nothing more, i.e., no information on the *secret bit* b_{1-c} is to be revealed.

Formally speaking, the ideal protocol implementing OT must satisfy the following properties:

- *Security for A:* If A is honest, then B gets no information on the secret bit b_{1-c}.
- *Security for B:* If B is honest then he receives b_c while A gets no information on c.

Similarly, in a Rabin Oblivious Transfer [Rab81], A sends a bit to B such that it is either received with some probability or completely lost otherwise. A does not find out which of the events took place, while B knows for sure whether he got the bit or did not. In communication theory, the Rabin OT is known as an *erasure channel*. Independently from [Wie83] but inspired by [Rab81], OT was introduced in [EGL83] with an application to contract signing protocols. These two flavours of OT tured out to be equivalent as shown in [Cré87].

The importance of OT is that it suffices to achieve secure two-party computations [GMW87] in a computational model as well as in an abstract (in particular, unconditional) model [Kil88, Cré89].

[1] In our work, we shall focus on this flavour of Oblivious Transfer. We refer to it throughout as just Oblivious Transfer.

In fact, the results of Kilian and also Beimel, Malkin and Micali [Kil91, BMM99, Kil00] show that any primitive computing a *non-trivial*[2] probabilistic function on the parties' inputs suffices to achieve multi-party computations. However, in unconditional model, it only holds for the case of passive adversaries.

A variety of implementations for an unconditional OT has been known since 1980-ies and the new ones has appeared in the recent years [CK88, Cré97, CCM98, DKS99, Riv99, NP00] proving the OT still to be a topical problem of high theoretical interest.

OT protocol based on Binary Symmetric Channel was first introduced by Crépeau and Kilian in [CK88] and later improved in [Cré97, KM01, SW02]. The reduction of OT to Unfair Noisy Channel was introduced in the work of Damgård, Kilian and Salvail [DKS99] with revised and extended possibility results given in [DFMS04].

1.3 Bit Commitment

The BC schemes was introduced by Chaum, Damgård and van de Graaf in [CDG87] as one of the building blocks for multi-party computation. Bit commitment scheme is a pair of protocols Commit and Open executed by two parties, a sender (or *committer*) A, and a receiver B. First, A and B execute Commit that results in B holding an evidence of A's input value b. Ideally, the receiver should learn no information about b from this. Later, A and B may execute Open after which B outputs "accept b", or "reject". Note that in principle, the opening phase may never happen. Formally, the ideal commitment scheme must satisfy the following properties:

- *Hiding:* If the sender A is honest, then committing to b reveals no information about b to B.
- *Binding:* If the receiver B is honest, then he always accepts with some value which is exactly the same that the honest A wants to commit to. Furthermore, A cannot open the commitment such that B accepts a different value.[3]

Commitment schemes are essential in the construction of a number of cryptographic protocols. General zero-knowledge proofs and arguments of [GMW86] and [BCC88] as well as multi-party computations of [CDG87] and [GMW87] are based on commitment schemes.

In the computational model, it is possible to build the bit commitment schemes which are unconditionally binding and computationally hiding (see, e.g., [Nao87]). At the same time, in [CDG87], there also shown schemes which are unconditionally hiding and computationally binding. Nevertheless, it is impossible to construct a commitment which is both unconditionally binding

[2] See [BMM99] for the definition of non-triviality.

[3] Basically, this property means that Alice cannot change her mind about the committed data.

and hiding, and based on noiseless communications without any additional assumption. The intuitive reason is the so-called *symmetry condition* (see Subsection 2.9.1) on what participants know about each other's data. Since both parties possess the entire transcript of the conversations that has taken place between them, in a two-party case, the player A can determine exactly what B knows about A's data, and the same holds for B.

One of the methods to break the symmetry condition was presented in [CK88] and later developed in [Cré97]. The idea was to construct the primitive based on noisy channels. The work [CK88] presented the reduction of Oblivious Transfer to Binary Symmetric Channel (BSC), while OT implies BC according to [Blu81]. In [Cré97], there was introduced a bit commitment protocol which was based directly on BSC. Finally, the bit commitment scheme based on a weaker (and more realistic) assumption of Unfair Noisy Channel was presented in [DKS99].

1.4 Noisy Communication Models

The protocols on which we focus in our work take place in a model with two players A and B connected by a noisy channel and an error-free channel in addition. The noise is considered to be a valuable resource since the security relies on the randomness provided by the channel. The noiseless communication is assumed to be given "for free".[4]

The *Binary Symmetric Channel (BSC)* is a well-studied (and the simplest) communication channel model: in this channel, every transmitted bit (independently of its value) is flipped with probability δ such that $0 \leq \delta \leq 1/2$, otherwise the bit is delivered with no error. The value δ is called the *error rate (or probability)* of the BSC. A BSC is called *memoryless* if every sent bit gets flipped independently on the other ones. We consider only the memoryless channels in this work. The obvious advantage of the BSC model is its simplicity however, this model is not a realistic formalisation of the process which takes place in the real communication channel. It is possible in principle that a dishonest player gains some control over the channel aiming to change the error rate. It can be done by transmitting the artificial noisy signal or enhancing the receiving equipment, for instance. An important observation is that it may be hard to hide the fact that the channel is *noisier* than it is prescribed. However, one can always hide the fact that one has made it *less noisy* by deliberately distorting the messages before/after transmission/reception.

The latter observations stipulate a necessity to introduce a new noise model capturing the possible cheating described above. We shall do that using so called *unfair noisy channels (UNC)* introduced in [DKS99]. In this model – which builds on the standard BSC – the error rate is guaranteed to fall into interval $[\gamma, \delta]$, while for each separate transmission the channel is in fact a BSC. The "unfairness" is due to the fact that the exact error probability is *not* known to the honest player but it may be set within the range of $[\gamma, \delta]$ by the cheating player.

[4] See the discussion in Subsection 2.9.2.

An adversarial behaviour can be *passive* (or *semi-honest*) and *active*. The latter means deviating from the protocol in any way: ignoring steps of the protocol, sending unprescribed data and so on. The passively cheating player is an "honest-but-curious" one. He follows the protocol, but tries to collect as much information about all the data sent and received as he can, and uses them to compute the other player's private data.

For example, in UNC scenario, the actively cheating party may set the error rate for every transmission to any value from the interval $[\gamma, \delta]$ he wishes. Whereas the passive adversary cannot affect the channel directly, he only can receive some additional information about the data transmitted, such that from his point of view the error rate reduces to some value in $[\gamma, \delta]$. We call this scenario a *Passive Unfair Noisy Channel* (*PassiveUNC*). Despite of the fact that the two scenarios look quite similar, there is however a crucial difference – in PassiveUNC, the corrupt player does not influence the actual error rate and the advantage exists *only from his point of view*.

1.5 Secure Protocol Composition

We consider *reductions* to be special protocols that use calls to one or more primitives in order to produce another protocols. When considering the security of reductions, it is quite important to show that several secure protocols which are combined or *composed* produce the new one which is in turn secure as well.

There have been invented a number of cryptographical tools whose task was to argue the security of the composed protocols which are build upon the other primitives (see, e.g., [Cré90, Bea91, MR91, PW00, Can01]). In our work, we make use of the *universally composable framework* of Canetti [Can01]. In this framework, players in a protocol can be given access to one or more *ideal functionalities*. Such a functionality can be thought of as a trusted party T with whom every player can communicate privately. There is a number of commands specified that T will execute. Every player can send a command to T, and T will faithfully carry out the command according to its specification, and may send results back to (some of) the players. Many cryptographic constructions – including ours – actually aim at building a protocol for the players only (without a trusted party) that does "the same thing" as some ideal functionality T, even if an adversary can corrupt either of the players and make them behave as he likes. The framework provides a precise definition of what it means that a protocol π in this way securely implements T. If this definition is satisfied, then any protocol that is secure when using T is also secure if T is replaced by π. In its full generality, the definition is robust against adaptive adversaries and concurrent composition of protocols.

All our protocols are in the two-player case with information-theoretic security. Here, the standard approach in previous research to security proofs has been to assume that either A or B is cheating, then prove some relevant security properties, and finally to prove that if both parties are honest, then the protocol "works correctly".[5] We express this in the UC framework by assuming

[5] In fact, this is the way we argue security of the constructions in Chapters 3 and 4. This

an infinitely powerful static (i.e., non-adaptive) adversary who from the start has corrupted no one, or either A or B.

Another consequence of being in the two-player case, is that we do not think of our protocols as subroutines in a multi-player protocol, nor are we worried about external observers, only about what a corrupted A or B might do or learn. We therefore assume that unless the adversary corrupts a player, he gets no information about the communication between A and B.

To prove that a protocol π satisfies the UC definition, one has to construct, for every adversary Adv attacking the protocol in question, an ideal model adversary, or *simulator* S, which gets to attack an ideal scenario where only the players and T are present. The goal of S is to achieve "the same" as Adv could have achieved by an attack on the real protocol. In the framework, this is formalised by assuming an *environment machine* Z which can communicate in a real-life attack with Adv and the honest players, and in the ideal model with S and the honest players. The protocol is said to be secure if for every adversary Adv there exists a simulator S, such that Z cannot distinguish whether it is in the real-life or the ideal model.

In proofs of this type of security, S usually works by running internally a copy of the adversary Adv, and passing interaction back and forth between Z and Adv with no change. If S can simulate with an indistinguishable distribution both the view of Adv attacking π and simultaneously make the input/output behaviour of the honest players be as in the real attack, then Z will not be able to tell any difference. We are in the unconditional model, so we demand a statistical indistinguishability for the aforementioned distributions. We also modify the UC model as given in [Can01] by allowing our adversaries and simulators infinite computing power – but we stress that honest players can execute our protocols efficiently. The noisy channels we study in this thesis can very conveniently be modelled as ideal functionalities, and reductions that build one type of channel from another can be proved secure in this framework.

1.6 Contributions of this Thesis

Our work which is based on [KM00,KM01,CMW04,DFMS04] is concerned with and contributes to the theory of unconditional secure primitives.

In Chapter 3, we present a protocol for Oblivious Transfer based on a Discrete Memoryless Channel with probability transition matrix of a general form. This result was previously published in [CMW04]. The complete characterisation of DMC with respect to OT is introduced since the presented construction is secure for any non-trivial DMC. The protocol builds on a famous result of Crépeau [Cré97] for OT based on a Binary Symmetric Channel, a particular case of DMC. We stress that our result gives a solution to the open question posed in [Cré97]: realising OT from any non-trivial BSC but in a more general

is justified by the fact that those protocols are essentially constructed directly from the noisy channel. In Chapter 6 on the contrary, our protocols build on a number of primitives.

way[6]. Moreover, the presented construction has an efficiency advantage[7] – we reduce the required number of channel uses from *cubic* to, roughly, *quadratic* order in $\log(1/p_f)$, where $p_f > 0$ is an upper bound on all the failure probabilities.

Some straight forward generalisations to the protocol for the case of BSC are presented and their asymptotic behaviour is analysed. Specifically, we consider the use of multiple repetition codes instead of a two-repetition code and conclude that the better efficiency can be obtained in the former case.

We conclude Chapter 3 with examining some special cases of DMC and discuss the corresponding simplifications which can be made to our protocol.

We stress that in Chapter 3, only the security in asymptotic case is considered, i.e., we allow the number of channel uses N to be large enough so that the failure probabilities become exponentially small in N. However, it is the performance in non-asymptotic case which is of practical importance. Hence in Chapter 4, we consider the security of Oblivious Transfer and Bit Commitment based on BSC [Cré97] in the non-asymptotic case. We derive relations for their failure probabilities depending on the number of channel uses and the lengths of error-correction codes involved which allows us to establish the trade-offs between the communication complexity on the one hand and the security on the other hand. This material was in part presented in [KM00, KM01]. In particular, we show that if one demands the failure probabilities for OT to be of order 10^{-6}, one needs to use the BCH error-correcting codes of length 65535 and the overall complexity of the protocol will be about 300 gigabytes. At the same time, Bit Commitment has significantly better performance for the same demands on failure probabilities with the BCH codes of length 16383 and overall complexity of a few kilobytes.

The protocols analysed in Chapters 3 and 4 were considered as stand-alone, their security was proven by arguing that they satisfy some particular properties. However, the modular approach is widely used in the protocol design where a protocol realises a particular task or a *functionality* for more advanced ones. Here, the notion of a *secure composition* for the protocols comes in handy. In Chapter 5, we consider the modification of Canetti's *Universally Composable (UC) framework* for the two-party unconditional model. We briefly argue that this extension is valid hereby preparing a ground for the results concerning OT based on Unfair Noisy Channels. We stress that our argument is somewhat informal, however it is one of the first attempts to adapt the UC framework for the case of unconditionally secure primitives. The preliminary ideas of this kind appeared in [Can00] for the case of secure function evaluation and in [BM04] for the quantum setting. At the same time, we do not use the full generality of the original framework restricting ourselves to the case of static adversaries.

The noise models considered in the previous chapters were somewhat restrictive since we always assumed that the players know the distribution of the noise *precisely*. On the contrary, in a more realistic model of Unfair Noisy Channels

[6]For the case of BSC, the solution was previously given in [KM01] and independently in [SW02].

[7]Originally, the idea comes from [SW02].

introduced in [DKS99], the adversary is given a partial control over the randomness introduced by the channel. The main question investigated in Chapter 6 is to what extent one can allow the adversary to control the channel such that OT still can be implemented. This material was published in [DFMS04]. A rather surprising result is that a passive adversary is almost as powerful as an adaptive one when constructing OT based on a UNC. This is shown by introducing a generic "compiler" which transforms any protocol implementing OT from a PassiveUNC and secure against passive cheating into a protocol that uses UNC for communications and builds an OT secure against active cheating. Here the results of the previous chapter come in handy, since we argue the new result in terms of the UC framework. This construction also fixes a flaw in the proof of [DKS99] which was incomplete otherwise. Finally, we exploit this compiler and a new technique for transforming between the weaker versions of OT, in order to prove a stronger possibility result then the one claimed in [DKS99]. In other words, there is now a much larger range of (γ, δ)-values for which one can implement OT based on a (γ, δ)-UNC. For instance, we can show that secure OT follows from a (γ, δ)-UNC with *any* value of δ between 0 and $1/2$, provided that γ is close enough to δ. This gives us the class of probabilistic primitives (UNC) which are complete for two-party secure computation in the sense of [Kil00].

Chapter 2

Preliminaries

This chapter contains some selected inequalities and theorems from probability theory, information theory, coding theory and cryptography. The material presented here is not self-contained and only aims at providing the theoretical background for the results presented in the next chapters. Comprehensive information on these topics can be found in the books by Feller [Fel68], Blahut [Bla87], Cover and Thomas [CT91], and MacWilliams and Sloan [MS77].

The logarithms denoted "log" are binary, the ones denoted "ln" are natural. The exponentiation function is $\exp(x) \equiv e^x$. The cardinality of a set $\mathcal{X}$ is denoted by $|\mathcal{X}|$.

2.1 Discrete Probability Theory

A *discrete probability space* consists of a finite or a countably infinite set Ω, the *sample space* together with a *probability distribution* P. The elements of the sample space Ω are called *elementary events*. Each elementary event can be viewed as a possible outcome of an experiment. We assume the reader to be familiar with basic properties of a probability distribution. Next, we introduce some known concepts and fix the notation.

Two events $\mathcal{A}$ and $\mathcal{B}$ are called *independent* if

$$P[\mathcal{A} \cap \mathcal{B}] = P[\mathcal{A}]P[\mathcal{B}]$$

The *conditional probability* $P[\mathcal{A}|\mathcal{B}]$ of an event $\mathcal{A}$ given that an event $\mathcal{B}$ occurs is defined as

$$P[\mathcal{A}|\mathcal{B}] = \frac{P[\mathcal{A} \cap \mathcal{B}]}{P[\mathcal{B}]},$$

where $P[\mathcal{B}] > 0$.

A *discrete random variable* X is a mapping from the sample space Ω to an alphabet $\mathcal{X}$. X assigns a value $x \in \mathcal{X}$ to each elementary event in Ω. The *probability distribution* of X is the function

$$P_X(x) = P[X = x].$$

The *conditional probability distribution* of a random variable X given an event $\mathcal{A}$ is defined as

$$P_{X|\mathcal{A}}(x) = P[X = x|\mathcal{A}].$$

The *joint probability distribution* of the random variables X and Y (with alphabet $\mathcal{X} \times \mathcal{Y}$) which are defined on the same sample space is denoted by $P_{XY}(x, y)$.

When conditioning on another random variable Y with the same sample space as X, the conditional probability distribution of X given that Y takes a value y is

$$P_{X|Y=y}(x) = \frac{P_{XY}(x, y)}{P_Y(y)},$$

where $P_Y(y) > 0$.

Two random variables are called *independent*, if for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$

$$P_{XY}(x, y) = P_X(x)P_Y(y).$$

The *expected value* of a discrete random variable X over $\mathcal{X}$ is

$$E(X) = \sum_{x \in \mathcal{X}} xP_X(x).$$

The *collision probability* of X is defined as the probability that X takes on the same value twice in two independent experiments:

$$P_c(X) = \sum_{x \in \mathcal{X}} P_X(x)^2.$$

The *Bernoulli trials* are defined as repeated independent trials if there are only two possible outcomes for each trial and their probabilities remain the same throughout the trials.

Let $X_1, \ldots, X_n$ be Bernoulli trials such that, for $1 \leq i \leq n$, $P[X_i = 1] = p$ and $P[X_i = 0] = 1 - p$ and $X = \sum_{i=1}^{n} X_i$; then X is said to have the *binomial distribution*.

More generally, let $X_1, \ldots, X_n$ be independent trials and, for $1 \leq i \leq n$, $P[X_i = 1] = p_i$ and $P[X_i = 0] = 1 - p_i$. Such trials are referred to as *Poisson trials*.

We introduce the so-called *Chernoff bounds* on the tail of the distribution of the sum $X = \sum_{i=1}^{n} X_i$ where X_i are Poisson trials, the bounds also apply to the special case where X_i are Bernoulli trials. The following two inequalities are proved in [MR95]:

Let $X_1, \ldots, X_n$ be Poisson trials as defined above. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = E(X) = \sum_{i=1}^{n} p_i$, and any $\delta > 0$,

$$P[X > (1 + \delta)\mu] < \left(\frac{\exp(\delta)}{(1 + \delta)^{(1+\delta)}} \right)^{\mu}, \tag{2.1}$$

$$P[X < (1 - \delta)\mu] < \exp\left(-\mu\delta^2/2\right). \tag{2.2}$$

The following more general result implied by Lemma 2.2.4 from [Rom90] will come in handy since it will allow us to estimate the tails *on both sides*.

Lemma 2.1 *Let $X_1, \ldots, X_n$ and μ be as defined above, then*

$$P\left[\frac{|X-\mu|}{n} > \nu\right] \leq 2\exp\left(-n\nu^2/2\right).$$

Note that for the case of Bernoulli trials (i.e., when $p_1 = p_2 = \ldots = p_n$), this reduces to the well-known *Bernstein's law of large numbers*.

2.2 Information Theory

The *(Shannon) entropy* of a random variable X with probability distribution P_X and alphabet $\mathcal{X}$ is defined as

$$H(X) = -\sum_{x\in\mathcal{X}} P_X(x)\log P_X(x)$$

with the convention that $0\log 0 = 0$, which is justified by the following fact: $\lim_{p\to 0} p\log\frac{1}{p} = 0$. The *conditional entropy* of X conditioned on a random variable Y is defined as follows:

$$H(X|Y) = \sum_{y\in\mathcal{Y}} P_Y(y)H(X|Y=y),$$

where $H(X|Y=y)$ denotes the entropy computed from the conditional probability distribution $P_{X|Y=y}$.

The entropy $H(X)$ of a random variable X is a measure of its *average uncertainty*. It is the minimum number of bits required on the average to describe the value x of the random variable X. Similarly, $H(X|Y)$ is the average number of bits required to describe X when Y is already known.

The probability distribution of a binary random variable is completely characterised by the parameter $p = P_X(0)$. The *binary entropy function* is defined as the entropy of such X, i.e.,

$$h(p) = -p\log p - (1-p)\log(1-p).$$

The *Rényi entropy of order two* (we shall refer to it as "Rényi entropy" throughout) of X [Rén61, Rén70] is defined as

$$R(X) = -\log P_c(X),$$

where $P_c(X)$ is the collision probability.

Analogously to the Shannon entropy, the conditional Rényi entropy is

$$R(X|Y) = \sum_{y\in\mathcal{Y}} P_Y(y)R(X|Y=y).$$

If X is a binary random variable, then:

$$R(X) = -\log\left(p^2 + (1-p)^2\right).$$

The rigorous study of Rényi entropy, its relation to the other information measures and their applications to cryptography can be found in [Cac97].

The *mutual information* of the variables X and Y is defined as

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P_{XY}(x,y)\log\frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}.$$

It is easy to verify that

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

In other words, $I(X;Y)$ can be thought of as the reduction of the uncertainty of X when Y is learned or the reduction of the uncertainty of Y when X is learned. It also follows that

$$I(X;Y) = I(Y;X).$$

Fano's inequality [Fan61] relates the probability of an error in guessing the random variable X to its conditional entropy $H(X|Y)$ where Y is a random variable correlated with X. Let $\hat{X}$ be an estimate of X. We define the probability of error in guessing as $p_e = P[\hat{X} \neq X]$, then

$$H(p_e) + p_e\log(|\mathcal{X}| - 1) \geq H(X|Y).$$

Let X be a random variable describing a uniformly distributed l-bit string b which is not known to an observer. Let Y be a random variable describing the result of some random experiment involving b. Then it is easy to verify that Fano's inequality transforms in this case into the following:

$$l + (1-p_c)\log\frac{1-p_c}{2^l-1} + p_c\log p_c \leq I(X;Y), \tag{2.3}$$

where $p_c = 1 - p_e$ is a probability for the observer to guess b correctly given Y.

2.3 Channel Capacity

Discrete channel is a system consisting of an input alphabet $\mathcal{X}$ and output alphabet $\mathcal{Y}$ and a probability transition matrix $P_{Y|X}$ that expresses the probability of observing the output symbol $y \in \mathcal{Y}$ given that the symbol $x \in \mathcal{X}$ has been sent. X and Y are the random variables which describe respectively, input and output of the channel. The channel is said to be *memoryless* if the probability distribution on the output depends only on the input at that time and is conditionally independent of previous channel inputs or outputs. We refer to *discrete memoryless channel* as *DMC*.

We define the *channel capacity* of a DMC as

$$C = \max_{P(x)} I(X;Y),$$

where the maximum is taken over all possible input distributions $P(x)$.

Binary Symmetric Channel (BSC) has binary input and output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and

$$P_{Y|X} = \begin{pmatrix} 1-\delta & \delta \\ \delta & 1-\delta \end{pmatrix},$$

i.e., it flips every transmitted bit with fixed *error probability* or *rate* δ, no matter what the value of the bit is. We denote this channel as δ-BSC. It is easy to show that the capacity of this channel is

$$C_p = 1 - h(p).$$

Cascading two BSC's with error rates μ and γ, respectively results in another BSC with error rate $\mu(1-\gamma) + \gamma(1-\mu)$. We denote

$$\mu(1-\gamma) + \gamma(1-\mu) \stackrel{def}{=} \mu \boxplus \gamma. \tag{2.4}$$

It is easy to verify that the operator "$\boxplus$" is commutative, associative and satisfies that if $|\mu - \mu'| < \delta$, then $|\mu \boxplus \gamma - \mu' \boxplus \gamma| < \delta$ for all γ.

An interesting non-binary channel is the *erasure channel* which has a binary input alphabet $\mathcal{X}$ and an output alphabet $\mathcal{Y} = \{0, 1, \Delta\}$. Here, the bits are either received without errors with probability p, or completely lost otherwise, in which case the channel outputs "Δ". The erasure channel has the following transition matrix:

$$P_{Y|X} = \begin{pmatrix} p & 0 & 1-p \\ 0 & p & 1-p \end{pmatrix}.$$

Finally, if the channel has some finite input alphabet $\mathcal{X} = \{x_1, \ldots, x_{|\mathcal{X}|}\}$ and finite output alphabet $\mathcal{Y} = \{y_1, \ldots, y_{|\mathcal{Y}|}\}$, then

$$P_{Y|X} = \begin{pmatrix} P_{Y|X=x_1}(y_1) & P_{Y|X=x_1}(y_2) & \ldots & P_{Y|X=x_1}(y_{|\mathcal{Y}|}) \\ P_{Y|X=x_2}(y_1) & P_{Y|X=x_2}(y_2) & \ldots & P_{Y|X=x_2}(y_{|\mathcal{Y}|}) \\ \cdots & \cdots & \cdots & \cdots \\ P_{Y|X=x_{|\mathcal{X}|}}(y_1) & P_{Y|X=x_{|\mathcal{X}|}}(y_2) & \ldots & P_{Y|X=x_{|\mathcal{X}|}}(y_{|\mathcal{Y}|}) \end{pmatrix}.$$

In general, there is no closed form solution for the capacity. Some methods for calculating it are discussed, e.g., in [CT91]. In our work, we always assume that the capacity of the channel in question is known.

2.4 Coding Theory

A *binary error-correcting code* C with codeword length or size n, dimension k and minimal distance d is a subset of cardinality 2^k of $\{0, 1\}^n$, the *codewords*, such that for any two elements $v, w \in C$, $d_H(v, w) \geq d$ holds, where "d_H" denotes the *Hamming distance* between two bit strings, i.e., the number of positions where they differ. Of particular importance is the special case of *linear codes*, where the subset of codewords is in fact a k-dimensional linear subspace of $\{0, 1\}^n$. In this case, the code is called a $[n, k, d]$-code and can be

represented by a $k \times n$ matrix $\mathbf{G}$, the *generating matrix*, or, alternatively, by the $n \times r$ *parity-check matrix* $\mathbf{H}$, where $r = n - k$ is a number of check bits in the code. We define a *Hamming weight* of a word v, $w_H(v)$ as a number of ones in it.

For our purposes, we need *asymptotically good* codes which are the codes that achieve a constant *rate* $R = k/n$ and *relative distance* d/n for arbitrary large n. The well-known fact ([MS77], Ch. 17, Prob. 31) is that (informally speaking) a randomly chosen generating matrix provides us with an asymptotically good code with high probability as long as the rate of the code is less than the channel capacity. Formally, the following theorem is true:

Theorem 2.1 *There exists a constant $\rho > 1$ such that a random binary matrix $\mathbf{G}$ of size $Rn \times n$ defines a binary linear code with minimal distance at least ϵn except with probability not greater than $\rho^{(R-C_\epsilon)n}$, for values $R < C_\epsilon$.*

However, we cannot use the random codes when *efficiently decodable* asymptotically good codes are needed since the decoding procedure for the random linear codes is not known to be efficient in general case. For this purpose, we make use of a special class of linear codes, so-called *concatenated codes* introduced by Forney [For66], these codes are both asymptotically good and efficiently decodable.

Theorem 2.2 *For any $\varphi > 0$ there exists $\rho > 1$ such that for all $R < 1 - h(\varphi)$ and sufficiently large N there exists a linear code with the length N and number of check bits at most $(1 - R)N$, failing to correct φN uniformly distributed errors only with probability at most $\rho^{(R-1+h(\varphi))N}$.*

We sketch the construction of concatenated codes [For66]. A straight forward Las Vegas construction algorithm combines a power-of-two ($N = 2^n$) size $[N, (1-\alpha)N, \alpha N - 1]$ *Extended Reed-Solomon (outer) code* over the field $\mathbf{F}_{2^n}$ to a rather good (inner) code of size n selected at random among all linear codes $[n, \kappa n, \delta n]$ of appropriate dimension κn. The resulting concatenated code has parameters $[Nn, (1-\alpha)\kappa Nn, \alpha\delta Nn]$ and is able to efficiently correct up to nearly δNn errors on average if they are uniformly distributed (because only very few errors will be uncorrected by the inner code). The error correction procedure uses a brute-force search for the nearest codeword on the inner code and the Berlekamp-Massey algorithm for the outer Extended Reed-Solomon code. Both of these algorithms run in polynomial-time with respect to the global code size Nn.

In some cases, the information transmitted will not be a codeword but only a *syndrome* $syn(r) = H^T r$, while the noisy versions of the information bits are already known to the receiver. From this syndrome, the decoding algorithm allows for recovering $r \in \{0,1\}^n$, given its noisy version $r' = r \oplus e$, where e is an error vector. In details, the sender is to announce the syndrome of r, $\mathbf{H}^T r$, so that the receiver can calculate $\mathbf{H}^T r' \oplus \mathbf{H}^T r = \mathbf{H}^T e$ (by linearity) and then, using the decoding algorithm, the coset leader $\hat{e}$. Clearly, $\hat{e} = e$, if the code is capable to correct errors in the noisy channel, so the receiver recovers r as $r' \oplus \hat{e} = r \oplus e \oplus e = r$.

2.5 Universal Hashing and Privacy Amplification

In information-theoretical cryptography, *hash functions* are often used for distribution uniformising or, more specifically, concentrating a corrupt player's uncertainty as a part of *privacy amplification* technique.[1] In this context, of particular importance are *universal hash functions* introduced by Carter and Wegman [CW79, WC81].

Definition 2.1 *A 2-universal class of hash functions is a set $\mathcal{G}$ of functions $\mathcal{X} \to \mathcal{Y}$ such that for all distinct $x_0, x_1 \in \mathcal{X}$, there are at most $|\mathcal{G}|/|\mathcal{Y}|$ functions $g \in \mathcal{G}$ such that $g(x_0) = g(x_1)$.*

The 2-universal hash functions will be further referred to as *universal hash functions*.

Privacy amplification was first proposed in the context of quantum key agreement for the special case of deterministic side information [BBR86] and later in general [BBCM95]. On the other hand, the effect of additional side information, in our case the syndrome the receiver learns, was studied in [BBCS92, BCJL93].

The basic idea is that if one knows some (linear) fraction of a bit-string, this knowledge can be reduced drastically once this string is hashed. In other words, if the hash function is applied to the string, then one knows much less than the linear fraction of the output in fact, only exponential fraction of it.

This works even if the partial information is a noisy version of the bit-string, transmitted over some channel. In some scenarios, an additional side information (such as parity check, for example) is available to the player allowing him to recover some (possible all) errors. Then, roughly speaking, the number of bits by which the resulting almost secret string will be shorter corresponds to the length of this side information.

Theorem 2.3 [BBCM95, BBCS92, BCJL93] *Let V be a uniformly distributed n-bit string and let W be generated by independently sending each bit of V over a φ-BSC. Let, furthermore, $syn : \{0,1\}^n \to \{0,1\}^r$ be a linear function and G be a random variable corresponding to the random choice, according to the uniform distribution, of a function from a universal class of hash functions*[2] *$\{0,1\}^n \to \{0,1\}^l$. Then,*

$$I(G(V); G, W, syn(V)) \leq 2^{-(R(W|V=v)-l-r)} / \ln 2,$$

for all sufficiently large n.

The following fact is proved in [BBCM95] for the setting described above: informally speaking, if we conceptually give away to the observer the Hamming distance between W and the particular value v, then for all sufficiently large n, $R(W|V = v)$ will be lower bounded by roughly $nh(\varphi)$ almost always.

[1] For the other classical result using hash functions see, e.g. [HILL91].

[2] For instance, G can be a random linear function mapping n bits to l bits.

Lemma 2.2 [BBCM95] *Consider the auxiliary variable $U = d_H(W, v)$ and let any fixed $\gamma > 0$ and $\delta > 0$ be so that $h(\varphi - \delta) > h(\varphi) - \gamma/3$, then*

$$R(W|U = u, V = v) > (h(\varphi) - \gamma/2)n$$

for all sufficiently large n, except with probability exponentially small in n.

2.6 Statistical Indistinguishability

A *distribution ensemble* $X = \{X(k,z)\}_{k\in N, z\in\{0,1\}^*}$ is an infinite set of probability distributions, where a distribution $X(k,z)$ is associated with each $k \in N$ and $z \in \{0,1\}^*$. We shall consider the ensembles which describe outputs where the parameter z represents input, and the parameter k is taken to be the *security parameter*. Let $X = \{X(k,z)\}_{k\in N, z\in\{0,1\}^*}$ and $Y = \{Y(k,z)\}_{k\in N, z\in\{0,1\}^*}$ be two ensembles of distributions over $\{0,1\}^*$. The notion of *statistical indistinguishability* of X and Y essentially expresses the fact that one can replace X and Y by one another, for k large enough, with respect to an observer who may have an unbounded computational power, but a limited number of sample points. Formally:

Definition 2.2 *The distribution ensembles X and Y are* statistically indistinguishable *(or* statistically close*), denoted by $X \simeq Y$, if for any $c > 0$ there exists $k_0 \in \mathbf{N}$ such that for all $k > k_0$ and for all z:*

$$\sum_{u\in\{0,1\}^*} \big|P[X(k,z) = u] - P[Y(k,z) = u]\big| < k^{-c}.$$

2.7 Zero-Knowledge Proofs

The general zero-knowledge (ZK) proofs introduced by Goldwasser, Micali and Rackoff in [GMR85] served the purpose of proving the possession of some data in a particular form (for instance, the Hamiltonian cycle of a graph) without disclosing any additional information on this data.

Definition 2.3 *An interactive proof system (P,V) for language L is* zero-knowledge *if for every verifier V^*, there is a simulator M_{V^*} such that for $x \in L$ and any auxiliary input z, the distribution of conversations output by M_{V^*} on input x, z is statistically close to the distribution of conversations produced by (P, V^*) on input x and z (given to V^*).*

This is the definition of so-called *statistical* zero-knowledge where the prover P and the verifier V are unlimited in computing power that is indeed the case in our unconditional setting. Roughly speaking, the definition says that if V knew that the statement was true then he would be able to generate a convincing proof himself with a distribution very close to the one which the real prover would produce.

It follows from the work of Ben-Or et al. [BGGHKMR88] that given a bit commitment scheme, one can always construct a new one, where one can prove in zero-knowledge that committed bits satisfy a given Boolean formula.

2.8 On Some Security Properties

The following security properties which are important in general case turn out to be irrelevant with respect to the two-party primitives of Oblivious Transfer and Bit Commitment considered in this work:

- *Fairness:* A corrupted player should receive his output if and only if the honest player also receive his output.

  Note that in both OT and BC primitives the output is defined only for one of the players. In other words, these primitives are designed for *transferring* rather than *exchanging* information. Hence, we do not consider this property in our work.

- *Guaranteed Output Delivery:* A corrupted player should not be able to prevent honest player from receiving their output.

  In two-party case, the corrupt player may refuse to continue interaction at any point thus violating this property and there may be no way to prevent it. Thus, we do not require our protocol to have this property. Instead, we shall require that the output is correct once the protocol is completed successfully. Furthermore, in our protocols, the private inputs of the honest players remain private even if the corrupt player aborts the protocol at any step.

2.9 On Communication Models

2.9.1 Symmetry Condition

The well-known fact is that no two-party primitive with unconditional security for both parties is possible based on noiseless communication (with no additional assumption) even if each party can make his own random coin flips and only security against passive adversary is required. The reason is the so-called *symmetry condition* on what the participants know about each other's data. Both parties possess the entire transcript of the conversations that took place between them. Thus, in a two-party protocol, A can determine exactly what B knows about A's inputs, and the same holds for B.

In particular, for the case of Bit Commitment, an informal argument proceeds as follows: Suppose that we have an unconditionally binding and hiding bit commitment. Now, if A commits to, e.g., value "0", then she must be able to open it as both "0" and "1", otherwise B (whose computational power assumed infinite) could figure out that the committed value is not "1", violating unconditional hiding. But then, it means that A is able to change her mind, violating unconditional binding.

The argument for Oblivious Transfer involves an unconditionally secure reduction of BC to OT [Kil88]. If an unconditionally secure OT based on noiseless channel was possible then so would be BC, a contradiction.

2.9.2 Availability of Noiseless Channel

In our work, we consider the noisy channel to be an important and expensive resource since unconditional security relies on the randomness provided by the channel. Therefore, we assume for the sake of simplicity that a bi-directional noiseless channel is available to both parties "for free". Whenever we refer to a *communication complexity* of a protocol, we mean *the number of noisy channel uses* made by the players.

At the same time, if the error-free channel is unavailable to the players then the well-known error-correction techniques can be used in order to cope with noise and provide error-free communication. In this case, a substantial increase in communication complexity may take place.

We fix some terminology to be used in this thesis. We shall refer to the use of a noisy channel by saying that the player *sends* some data while using a noiseless channel, the player *announces* the data.

2.9.3 One-Directional Noisy Channel

Note that as matter of fact, Binary Symmetric Channel is symmetrical with respect to the players as well: if a δ-BSC is only available from A to B, then the following protocol allows to provide a δ-BSC in the reverse direction. Suppose B has an input bit b.

Protocol 2.1 δ-ReversedBSC(b)

1. A picks $r \in_R \{0,1\}$ and sends it to B who receives r'.
2. B announces $b' = b \oplus r'$ to A.
3. A outputs $b' \oplus r$.

It is easy to see that the same protocol works for a general DMC and (γ, δ)-UNC. This allows us to assume without loss of generality that the players are always connected by bi-directional noisy channels.

Chapter 3

Standard Assumption: Discrete Memoryless Channel

3.1 Introduction

The material of this chapter concerns a protocol for Oblivious Transfer based on Discrete Memoryless Channel. In our protocol, we shall use the same ideas as the ones presented by Crépeau in [Cré97] for implementing OT based on BSC. Essentially, we adapt their construction for the case of a DMC with a probability transition matrix of the general form. Some DMC's turn out to be "useless" or *trivial* with respect to obtaining OT from them. We define this notion of triviality and then focus on the case of non-trivial DMC's, since the famous impossibility result discussed in Section 2.9.1 immediately carries over to the trivial channels. We shall show that on the other hand, any non-trivial DMC *does* allow for realising OT. Non-triviality is, therefore, a necessary and sufficient condition for a possibility of unconditional OT based on DMC.

We first give the formal security definition for OT. Then, we discuss the triviality condition, introduce the construction and argue that it satisfies the claimed properties. Finally, some interesting special cases are considered.

This chapter is based on [CMW04, KM01].

3.2 Preliminaries

3.2.1 Some Notation

In this chapter, we refer to a value which is exponentially small in some security parameter n as to *negligible* in n. Whenever we refer to an event which occurs with *small* or *high probability* in n, we mean that the event occurs, respectively, with *probability negligible* or *except with probability negligible* in n.

3.2.2 Communication Model

We assume that a sender A and receiver B are connected by a DMC with the probability transition matrix $P_{Y|X}$ as well as a noiseless channel. Both A and B can make their local random coin flips.

3.2.3 Security Definition

In Section 1.2, an abstract definition of OT was given. We shall establish here explicitly what properties we achieve in the given construction where, we remind, the sender A transfers two bits b_0 and b_1 to the receiver B who gets only one of them, b_c, according to his selection bit c, while A learns nothing.

We note that the definition below builds on the standard one in order to capture an essential feature of some protocols (in particular, of our protocol) – the absence of the so called *input awareness* property. Input awareness demands that the players always behave consistently with some legal input values. Informally speaking, they must not be able to input a value which is unknown to them. We denote such the input as "?". We stress that the absence of this property does not constitute a flaw of the protocol, since there are standard methods which allow to gain the input awareness. See Subsection 3.5.1 for the discussion.

We define a function $f : \{View_Q\} \to \{(x_1, \ldots, x_N), ?\}$, where $Q \in \{A, B\}$ is a player, $View_Q$ is an encoding of Q's view of the protocol[1] and $(x_1, \ldots, x_N)$, a set of Q's inputs. Intuitively, the function is to tell whether the player behaves consistently with some inputs $(x_1, \ldots, x_N)$ or does not. If f outputs "?", we are not going to put any requirements on the output distribution – this is the way to state security properties without input awareness. We stress that the function f is used for the analysis only, the players are not to be aware of it.

Formally, we require the following properties:

- Completeness: if A and B are both honest, then they accept the protocol with high probability.

- Security for A: If A is honest, then $f(View_A) = (b_0, b_1)$ and with high probability, the distribution of at least one of the secrets b_0, b_1 is statistically close to uniform[2] (we shall call it *almost uniform*) over all possible $View_B$. Furthermore, if $f(View_B) = c$, then it is the distribution of b_{1-c} which is almost uniform[3]. Formally: If $f(View_B) = c$ then for all $v \in View_B$ except with negligible probability:
  $\big|P[b_{1-c} = 0 | View_B = v] - P[b_{1-c} = 1 | View_B = v]\big|$ is negligible.
  If $f(View_B) =?$ then there exists $i \in \{0, 1\}$ such that for all $v \in View_B$ except with negligible probability:
  $\big|P[b_i = 0 | View_B = v] - P[b_i = 1 | View_B = v]\big|$ is negligible.

- Security for B: If B is honest, then:
  $P\left[(f(View_A) = (b_0, b_1)) \text{ and } (B \text{ accepts}) \text{ and } (B\text{'s output} \neq b_c)\right]$ is negligible, and $D_{c=0}(View_A) \simeq D_{c=1}(View_A)$ where $D_{c=\{0,1\}}(\cdot)$ is the distribution over all possible A's views given B's selection bit c.

[1] It is defined as the input, all messages sent and received, and random coins used.

[2] We assume that b_0, b_1 are drawn uniformly by A.

[3] This slight complication in our requirements comes from the observation that the corrupt B may cheat passively and hence, he must loose the secret bit b_{1-c} according to the original OT definition. At the same time, when B behaves inconsistently, we want him to lose at least one of A's inputs, such that A's privacy will still be in place.

3.2.4 Binary-Symmetric Erasure Channel

Binary-Symmetric Erasure Channel (denoted (φ, ε)-BSEC) is yet another model of noisy communication – a generalisation combining BSC and erasure channel – which we are going to use in this chapter. It is a binary-input and ternary-output channel where an input symbol is erased with probability ε and each non-erased symbol has an error-rate φ. These two probabilities are the same for both input values.

3.2.5 Alternative Representation

An alternative representation of the DMC $P_{Y|X}$ is a set of points of $\mathbf{R}^{|\mathcal{Y}|-1}$ such that each point represents an input symbol. The coordinates of the point are the probabilities for this input to be received as the corresponding output symbol. Formally, for any $i = 1, \ldots, |\mathcal{X}| - 1$:

$$x_i = \left(P_{Y|X=x_i}(y_1), P_{Y|X=x_i}(y_2), \ldots, P_{Y|X=x_i}(y_{|\mathcal{Y}|-1})\right) \in \mathbf{R}^{|\mathcal{Y}|-1}.$$

Note that for every $x_i \in \mathcal{X}$: $\sum\limits_{y \in \mathcal{Y}} P_{Y|X=x_i}(y) = 1$.

3.3 Trivial Versus Non-Trivial Discrete Memoryless Channels

We first define a *redundant* input symbol of the DMC as a symbol whose output distribution can be expressed as a convex linear combination of the output distributions of the other inputs. Throughout this chapter, we refer to the output distribution of an input symbol x as $P_{Y|X=x}$ instead of the formal "$P_{Y|X=x}(y)$ for all $y \in \mathcal{Y}$".

Definition 3.1 *Let $P_{Y|X}$ be a DMC. We call an input symbol $x \in \mathcal{X}$* redundant *if its output distribution $P_{Y|X=x}$ can be written as follows:*

$$P_{Y|X=x} = \sum_{x' \in \mathcal{X} \setminus \{x\}} \mu_{x'} P_{Y|X=x'}$$

with $\mu_{x'} \in [0, 1]$, $\sum\limits_{x'} \mu_{x'} = 1$.

Lemma 3.1 *If Protocol 3.3 is (asymptotically) secure against an active adversary who uses only non-redundant input symbols then the reduction is also secure against an active adversary who uses all possible input symbols.*

Proof. (Sketch) In Protocol 3.3, it is only the sender A who uses the DMC for communication. The sender is instructed to use only the non-redundant symbols, so if A is honest and B is dishonest then the proof trivially follows.

Suppose now that A is corrupt and B is honest. In the asymptotic case, any possible output distribution which is generated using all input symbols can also be generated using only non-redundant symbols according to Definition 3.1.

Therefore, if there exists a strategy for A to violate the security of the protocol using some symbols including redundant ones (and so a certain output distribution corresponding to this strategy), then clearly there exists a strategy which achieves the same goal (and simulates the same output distribution) using only non-redundant inputs. The claim now follows. □

Yet another representation of a DMC comes in handy in order to make the intuitive definition of its triviality. Let us represent the DMC as a bipartite graph where the vertices of one set are inputs x and the vertices of the other one are outputs y. The vertices x and y are adjacent whenever $P_{Y|X=x}(y) > 0$. We call a DMC *trivial*, if one can partition the corresponding graph into the graphs which represent the channels each with capacity zero or one. In other words, if for every output symbol, one either can learn the input for sure or has no information on the output at all.

Definition 3.2 *We call a channel $P_{Y|X}$* trivial *if there exist, after removal of all redundant input symbols, partitions $\mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_N$, $\mathcal{Y} = \mathcal{Y}_1 \cup \cdots \cup \mathcal{Y}_N$, and channels $P_{Y_i|X_i}$, where the ranges of X_i and Y_i are $\mathcal{X}_i$ and $\mathcal{Y}_i$, respectively, such that*

$$P_{Y|X=x}(y) = \begin{cases} P_{Y_i|X_i=x}(y) & \text{if } x \in \mathcal{X}_i,\ y \in \mathcal{Y}_i,\ i = j \\ 0 & \text{if } x \in \mathcal{X}_i,\ y \in \mathcal{Y}_j,\ i \neq j \end{cases}$$

holds and such that the capacities of these channels $\forall i : C(P_{Y_i|X_i}) \in \{0, 1\}$.

Clearly, this is another way of saying that the players are connected by either an error-free channel (capacity 1 per bit) or disconnected completely (capacity 0), so that the impossibility result discussed in Subsection 2.9.1 follows immediately.

We are going to show next that the non-triviality of the channel implies its certain properties, namely the existence of two particular input symbols x_1, x_2 which, loosely speaking, cannot be simulated by any combination of the other inputs. As we show later, the consequence is that the sender A can be forced to use only those two symbols, since her failure to do so would be detected by the receiver B. As a matter of fact, this restriction is not going to work for the symbols which can be expressed as a linear combination of x_1, x_2. However, those ones are redundant and hence, they can be safely ignored.

Theorem 3.1 *Let $P_{Y|X}$ be a non-trivial channel. Then there exist $x_1, x_2 \in \mathcal{X}$ with the following properties:*

1. *$P_{Y|X=x_1} \neq P_{Y|X=x_2}$.*
2. *There exists $y \in \mathcal{Y}$ such that $P_{Y|X=x_1}(y) > 0$ and $P_{Y|X=x_2}(y) > 0$.*
3. *Let, for $\lambda, \mu_i \in [0, 1]$,*

$$\lambda P_{Y|X=x_1} + (1 - \lambda) P_{Y|X=x_2} = \sum_i \mu_i P_{Y|X=x_i}.$$

Then $\mu_i > 0$ implies that $P_{Y|X=x_i} = \tau P_{Y|X=x_1} + (1 - \tau) P_{Y|X=x_2}$ holds for some $\tau \in [0, 1]$.

Proof. By the non-triviality of the channel, there exist two non-redundant input symbols x_1 and x_2' and $y \in \mathcal{Y}$ such that $P_{Y|X=x_1} \neq P_{Y|X=x_2'}$, $P_{Y|X=x_1}(y) > 0$, and $P_{Y|X=x_2'}(y) > 0$ hold.

Recall the alternative representation of $P_{Y|X}$ given in Subsection 3.2.5. In the following, we will consider the *convex hull* of the set of all inputs

$$\{x|\, x \in \mathcal{X}\} \subseteq \mathbf{R}^{|\mathcal{Y}|-1}. \tag{3.1}$$

We call x_0 a *spanning point* of the convex hull if the convex hull of $\{x, |\, x \in \mathcal{X} \setminus \{x_0\}\}$ is strictly smaller than the one of (3.1).

Since the spanning points of the hull correspond to non-redundant inputs, we can conclude that there exist two spanning points x_1 and x_2' of the convex hull such that there exists $y \in \mathcal{Y}$ with $P_{Y|X=x_1}(y) > 0$ and $P_{Y|X=x_2'}(y) > 0$.

Let v_x be the unit vector parallel to the vector in $\mathbf{R}^{|\mathcal{Y}|-1}$ which connects x_1 with some x in $\mathcal{X}$. In a similar way as for the points, we define convex linear combinations and non-redundancy for these vectors. We say that the vector is *non-redundant* if it cannot be expressed as a convex linear combination of the other vectors. Let $\mathcal{A} = \{x \in \mathcal{X} | v_x \text{ is a non-redundant vector}\}$. In the following discussion, whenever we refer to *linear combinations*, we mean the *convex* ones.

The set of spanning points (to which x_1 belongs), can be thought of as the set of vertices of the polytope in $\mathbf{R}^{|\mathcal{Y}|-1}$. Then, intuitively, $\mathcal{A}$ is the set of vertices "adjacent" to x_1. They are adjacent in the sense that the lines (in $\mathbf{R}^{|\mathcal{Y}|-1}$) connecting x_1 with any $x \in \mathcal{A}$ do not go "through" the polytope. Observe that all the linear combinations of x_1 and some $x_2 \in \mathcal{A}$ belong to the line in $\mathbf{R}^{|\mathcal{Y}|-1}$ connecting x_1 with x_2. Then, it is enough to argue that $P_{Y|X=x_2}(y) > 0$.

Assume that for all $x \in \mathcal{A}$, we have $P_{Y|X=x}(y) = 0$. Then the same is true also for all distributions in the convex hull of these points. On the other hand, the line connecting x_1 with x_2' has a non-empty intersection with this convex hull by definition of $\mathcal{A}$. Since every distribution in this intersection is a linear combination of $P_{Y|X=x_1}$ and $P_{Y|X=x_2'}$ – both non-zero in y – there exists a point x_2 in $\mathcal{A}$ with $P_{Y|X=x_2}(y) > 0$.

By construction, x_1 and x_2 have now the following properties. First, they satisfy $P_{Y|X=x_1} > 0$ and $P_{Y|X=x_2} > 0$. Second, any linear combination of x_1 and x_2 cannot be represented as a linear combination involving points x *not* lying on the line connecting x_1 with x_2. This observation concludes the proof. □

3.4 Protocol

Our protocol is an adaptation of the protocol of [Cré97] for the general case where, at the same time, we reduce the required number of channel uses from *cubic* to, roughly, *quadratic* order in $\log(1/p_f)$ where $p_f > 0$ is an upper bound on all the failure probabilities. We develop the protocol in three steps. In Subsection 3.4.1, the original DMC is used to obtain a binary-symmetric erasure channel with error; in Subsection 3.4.2, this is transformed into a passively secure form of OT (which is vulnerable to active attacks by the sender A);

in Subsection 3.4.3, we introduce the final protocol avoiding these attacks by statistical analysis on the receiver's side.

3.4.1 Implementing Binary-Symmetric Erasure Channel

From a non-trivial channel $P_{Y|X}$, we first construct a BSEC. We encode the bits to be transmitted over the DMC as pairs of two fixed distinct input symbols $x_1, x_2 \in \mathcal{X}$ chosen according to Theorem 3.1. We proceed as follows: "0" is encoded as x_1x_2 and "1" as x_2x_1. When this is repeated many times, B gets the bits with different error rates which depend on the actual output symbols received. We will have B make a decision on 0 or 1 only when he receives certain specific pairs; otherwise, he will decide on erasure Δ. More precisely, B will accept only the pairs which give him the best estimate of what has been sent; we shall call those the *most informative pairs*. In general, there might even be output symbols y which allow for deciding *with certainty* whether x_1 or x_2 has been sent. Note, however, that the choice of x_1 and x_2 *guarantees* that there exist pairs which are *not* conclusive with certainty. The crucial point is that there are at least two different levels of conclusiveness, and it is the difference between the two that will be used in the protocol. In the following, we shall call the most informative pairs the *good pairs* for short and denote them as y_1y_2 and y_2y_1, respectively.

In the following protocol, the sender A has an input bit r. Let x_1, x_2 be the input symbols chosen according to Theorem 3.1. A is assumed to be passively cheating, if she is corrupt.

Protocol 3.1 $P_{Y|X} \to \mathrm{BSEC}(r)$

1. A sends x_1x_2 if $r = 0$ and x_2x_1 if $r = 1$.
2. B outputs $\begin{cases} 0 & \text{if } y_1y_2 \text{ is received,} \\ 1 & \text{if } y_2y_1 \text{ is received,} \\ \Delta & \text{if any other pair is received.} \end{cases}$

Let $\mathcal{Y}'$ be the set of y with $P_{Y|X=x_1}(y) > 0$ or $P_{Y|X=x_2}(y) > 0$.

Lemma 3.2 *A* most informative pair (y_1, y_2) *is the pair* $(y, \bar{y}) \in \mathcal{Y}' \times \mathcal{Y}'$, $y \neq \bar{y}$, *that achieves the following minimum:*

$$\varphi = \min_{(y,\bar{y}) \in \mathcal{Y}' \times \mathcal{Y}'} \frac{P_{Y|X=x_1}(\bar{y}) P_{Y|X=x_2}(y)}{P_{Y|X=x_1}(\bar{y}) P_{Y|X=x_2}(y) + P_{Y|X=x_1}(y) P_{Y|X=x_2}(\bar{y})} \,. \tag{3.2}$$

Proof. It is easy to see that Equation 3.2 indeed captures the intuitive definition of the most informative pair given above. A is passive, therefore she actually uses the encoding described in Protocol 3.1. Then the probability for B to accept a good pair is $P_{Y|X=x_1}(\bar{y}) P_{Y|X=x_2}(y) + P_{Y|X=x_1}(y) P_{Y|X=x_2}(\bar{y})$ while the error rate for the good pairs is $P_{Y|X=x_1}(\bar{y}) P_{Y|X=x_2}(y)$. Equation 3.2 describes the minimal such probability for all pairs of output symbols. □

Remark 3.1 The assignment of the inputs x_1, x_2 to the outputs y_1, y_2 is significant since it influences the error rate according to (3.2). We assume w.l.o.g. that x_1 is assigned to y_1 and x_2 to y_2.

Corollary 3.1 *The probability for B to accept a good pair in Protocol 3.1 is the following:*

$$p_g = P_{Y|X=x_1}(y_1)P_{Y|X=x_2}(y_2) + P_{Y|X=x_2}(y_1)P_{Y|X=x_1}(y_2). \tag{3.3}$$

Corollary 3.2 *Note that (3.2) is symmetric with respect to x_1 and x_2. The resulting channel looks, hence, like a $(\varphi, 1-p_g)$-Binary-Symmetric Erasure Channel. It is not exactly BSEC because the symbol Δ may contain some information about the input, but we shall treat this channel as such for the simplicity sake until more rigorous analysis in the next subsection.*

Remark 3.2 In general, there may exist several possible pairs of output symbols whose error rates are exactly φ. As B will later handle them according to their error rates, these pairs are virtually indistinguishable for him. Therefore, for the sake of simplicity, we shall later call all these pairs the *good* ones and refer to all of them as y_1y_2 and y_2y_1.

Once we defined the symbols x_1, x_2, y_1 and y_2, we shall consider only a partition $P_{Y_i|X_i}$ (see Definition 3.2) to which these symbols belong. Obviously, if the cheater uses the symbols from the other partitions, this will be detected with certainty. In the rest of this chapter, we refer to $P_{Y_i|X_i}$ as the DMC $P_{Y|X}$ connecting the players.

3.4.2 Passively Secure OT

The BSEC obtained above is not a Rabin OT: B might get some information even when deciding on Δ, and there are bit errors. We now describe a protocol, based on the obtained BSEC, for realizing OT under the assumption that A can cheat only passively. The general idea is to use the reduction from Rabin to One-out-of-two OT introduced in [Cré87].

In our passively secure OT protocol, A sends $2n$ random bits $r_1, r_2, \ldots, r_{2n}$ to B using BSEC. B should receive roughly $2p_gn$ of them as *good pairs* and $2(1-p_g)n$ as "bad" ones.

B then forms two sets I_0 and I_1, each of size $n' = n$ if $p_g > 1/2$ and, otherwise, of size $n' = (p_g + \beta)n$, $0 < \beta < p_g$. Through the index sets I_0 and I_1, B defines two bit-strings r'_{I_0}, r'_{I_1} such that r'_{I_c} – which corresponds to his selection bit – contains only good pairs. Accordingly, we define $p'_g = p_g$ if $p_g > 1/2$ and $p'_g = p_g/(p_g + \beta)$ otherwise.

Remember that φ is the error probability of the BSEC. The players now agree on an error-correcting code, according to the discussion in Subsection 2.4, which allows for correcting all errors in a set consisting only of good pairs. More precisely, the errors are corrected by having A send the syndromes $syn(r_{I_0})$ and $syn(r_{I_1})$. Using r'_{I_c} and $syn(r_{I_c})$, B can recover r_{I_c} except with probability negligible in n. On the other hand, this correction information is *not* sufficient

to find out *both* words r_{I_c} and $r_{I_{1-c}}$ with certainty, if the dimension of the code is equal to $n'\big((1-h(\varphi))(1+p'_g)/2 + (1-h(\varphi'))(1-p'_g)/2\big)$. Here, φ' is the error rate of a second most informative pair which is defined as the pair with the second lowest a posteriori error probability from B's viewpoint.

Finally, a privacy amplification function is used to extract one bit per string, such that one of the two bits may be recovered, but not both. This function is the scalar product (we denote it by "$\odot$") with a random bit string m of an appropriate length.

Protocol 3.2 BSEC $\rightarrow \widehat{\mathrm{OT}}(b_0, b_1)(c)$

1. A picks $2n$ random bits r_i, $i = 1, \ldots, 2n$, and sends them to B as BSEC(r_i); B receives r'_i.
2. B picks and announces to A two disjoint sets I_0, I_1, $|I_0| = |I_1| = n'$, such that $r'_i \neq \Delta$ holds for all $i \in I_c$.
3. A and B agree on a parity-check matrix $\mathbf{H}$ of a concatenated code C with parameters $[n', k = n'\big((1-h(\varphi))(1+p'_g)/2 + (1-h(\varphi'))(1-p'_g)/2\big), d]$ correcting $\psi\varphi n'$ errors, $\psi > 1$.
4. (a) A computes and sends $s_0 = syn(r_{I_0})$ and $s_1 = syn(r_{I_1})$,
   (b) picks and sends a random n'-bit word m, and
   (c) computes and sends $\hat{b}_0 = b_0 \oplus (m \odot r_{I_0})$ and $\hat{b}_1 = b_1 \oplus (m \odot r_{I_1})$.
5. (a) B recovers r_{I_c} using r'_{I_c}, s_c and the decoding algorithm of C and
   (b) computes and outputs $\hat{b}_c \oplus (m \odot r_{I_c})$.

The next lemma which comes in handy later follows from Lemma 2.1:

Lemma 3.3 *The probability that more than $\psi\varphi n'$ errors occur in r_{I_c} is negligible in n for any $\psi > 1$ and n large enough.*

Now, we argue the security of Protocol 3.2.

Proposition 3.1 *Assuming that the sender A is passively cheating, Protocol 3.2 implements OT with failure probabilities negligible in n.*

Proof. We argue the security for B first. The honest B must be able to receive his choice bit b_c except with probability negligible in n. It follows from the choice of the code C that B can correct $\psi\varphi n'$ errors with high probability in the string r_{I_c}. Observe that if $n' = n$ and $p_g \leq 1/2$ than the above does not hold, since B cannot completely fill the good set with good pairs[4]. Therefore, the size of the index sets I_0 and I_1 must depend on p_g. The choice of n' provides "scaling-down" of the index sets' size for the case when $p_g \leq 1/2$ and the new fraction of good bits p'_g is always greater than $1/2$. Now, Lemma 3.3 assures that B indeed corrects the errors in r_{I_c} almost always. Taking into account

[4]In case of $p_g = 1/2$, it does not happen with high probability.

that A is assumed to be honest and hence $f(View_A) = (b_0, b_1)$ always holds, we conclude that the honest B receives b_c with high probability.

The fact that the passively cheating A cannot learn c follows from the symmetry of the BSEC from A's point of view as noted in Corollary 3.2. The passive A can only find out B's selection bit by trying to tell the good set, r_{I_c} from the bad one, $r_{I_{1-c}}$. Clearly, she cannot do it better than guessing randomly for every input bit, no matter what B's input c is. The security definition for B is therefore satisfied.

A's security fails, if the corrupt B gets some non-negligible information about *both* bits b_c and b_{1-c} in other words, the corrupt B may not necessarily try to obtain b_c with certainty but attempt to gain at least some non-negligible information on both b_0 and b_1. We show that if B is cheating actively using his optimal strategy, then for at least one of the two bits his advantage in learning it – compared to a random guess – is exponentially small in n.

Note that the code is chosen such that *complete* error correction is possible only when B collects *all* the good pairs into one of the two sets. Suppose first that $p_g > 1/2$ holds so that $p'_g = p_g$. Then, there exists a constant fraction of bad bits whose error rate is at least $\varphi' > \varphi$, since φ' is the error rate for the second most informative pair. Next, we make two conservative assumptions for the sake of simplicity of our proof. First, we suppose that the fraction of the second most informative pairs is $1 - p'_g$ that makes the situation only better for the dishonest B. Second, we observe that all the adversarial strategies of B is about distributing the output pairs over the two sets, so we define the set with maximal number of the good pairs as the set corresponding to B's choice c and suppose that the dishonest B gets the bits b_c for free. It does not violate A's security since the honest B is entitled to get b_c anyway, but it makes our argument easier. It is now clear that given the above assumptions, the best strategy for the corrupt B, if he wants to learn as much as possible about both bits, is to split the good bits evenly over the sets I_0 and I_1. Therefore, the dishonest B is not able to put more than $p'_g n'$ good bits in at least one of the sets I_0 and I_1. The bits of this set do not contain more than $n'\big((1-h(\varphi))p'_g + (1-h(\varphi'))(1-p'_g)\big)$ bits of Shannon information about the original string with high probability. In fact, the entropy of the noisy versions of r_{I_0} and r_{I_1} is to be described by the Rényi entropy, however it follows from Lemma 2.2 that in our (asymptotic) scenario it is lower bounded by the Shannon entropy. Hence, the mutual information between r_{I_0}, r_{I_1} and their respective noisy versions can be upper bounded using standard Shannon information. Therefore, at least $n'\big(h(\varphi)p'_g + h(\varphi')(1-p'_g)\big)$ parity-check bits are needed in order to correct all the errors in each set with high probability; however, $syn(r_{I_0})$, $syn(r_{I_1})$ each contain at most $n'\big(h(\varphi)(1+p'_g)/2 + h(\varphi')(1-p'_g)/2\big)$ bits only. Thus, at least one of the two words r_{I_0}, r_{I_1} will be undetermined by at least $n'\big(h(\varphi') - h(\varphi)\big)(1-p'_g)/2$ bits where the terms in brackets are positive and do not depend on n. Now, Theorem 2.3 implies that after privacy amplification, B only has a negligible amount of information about the corresponding bit. Then, it follows from Fano's inequality (2.3) that B's advantage in guessing this bit will be negligible as well. It is easy to verify that the case of $p_g \leq 1/2$ follows by the same argument as above given the choices of n' and p'_g. □

Unfortunately, Protocol 3.2 is not secure against active cheating by A with the objective of figuring out B's selection bit c. For instance, A can send *incorrect pairs:* x_1x_1 or x_2x_2 instead of x_1x_2 and x_2x_1, hereby increasing the probability that they are received as bad pairs by B. Alternatively, A can use any other input symbols but x_1 and x_2 (we shall call them the *forbidden* input symbols) whose support intersects with those of x_1 and x_2. Finally, she can send an incorrect syndrome at Step 4.

In the first and second active attacks, the incorrect pairs are more likely to end up in the "bad" set, thus indicating to A which one of I_0 and I_1 is more likely to be the "good" and the "bad" set, respectively, and hence what B's choice is. In the third attack, if A renders only one of the syndromes incorrect, then B may abort the protocol or not, hereby disclosing which bit he is trying to get.

3.4.3 The Complete OT Protocol

The main idea is now, as in [Cré97], to avoid A's cheating by repeating Protocol 3.2 many times in such a way that A has to cheat in a substantial fraction of all executions of BSEC (namely, in more than the square root of the total number of executions) in order to gain useful information. This, however, can be detected by B when he analyses his output statistically.

More precisely, Protocol 3.2 is repeated $\lceil n^{1+\varepsilon} \rceil$ times, $0 < \varepsilon < 1$; thus, we apply BSEC $2\lceil n^{2+\varepsilon} \rceil$ times in total. In order to cheat, A will have to send at least $\lceil n^{1+\varepsilon} \rceil$ wrong pairs (i.e., she forms the pair incorrectly or uses forbidden symbols) in these executions. This will, however, lead to a detectable bias in the output distribution (with probability almost 1). If, on the other hand, A uses *less than* $\lceil n^{1+\varepsilon} \rceil$ incorrect pairs, she finds out nothing about c. Similarly, if A sends wrong syndromes in Protocol 3.2 she will, each time, be detected by B with probability 1/2. If she uses $n^{1+\varepsilon}$ such faulty syndromes it is, hence, only with negligible probability that B will not detect her cheating.

Let $n_\varepsilon = \left[n^{1+\varepsilon}\right]$, where $[\cdot]$ means rounding up to the next odd integer, and $n'_\varepsilon = n \cdot n_\varepsilon$. The instances are combined by requesting

$$b_{l,0} \oplus b_{l,1} = b_0 \oplus b_1, \ 1 \le l \le n_\varepsilon. \tag{3.4}$$

Let

$$b_{0,0} = \bigoplus_{l=1}^{n_\varepsilon} b_{l,0} \quad \text{and} \quad b_{0,1} = \bigoplus_{l=1}^{n_\varepsilon} b_{l,1} \ . \tag{3.5}$$

Then it is easy to verify that for some $c_l \in \{0,1\}$, $1 \le l \le n_\varepsilon$,

$$\bigoplus_{l=1}^{n_\varepsilon} b_{l,c_l} = b_{0,z} \quad \text{for} \quad z = \bigoplus_{l=1}^{n_\varepsilon} c_l \ . \tag{3.6}$$

Thus, in order to find out which of $b_{0,0}$ or $b_{0,1}$ B is trying to receive, A must find out *all* the c_l.

Let $\psi > 1$ be the one chosen in Protocol 3.2. We denote Protocol 3.2 where A has inputs $(b_{l,0}, b_{l,1})$ and B has input c_l by $\widehat{\mathrm{OT}}(b_{l,0}, b_{l,1})(c_l)$. An extra index

l is added to each variable of the lth iteration of $\widehat{\mathrm{OT}}$. Let us denote by $q_{l,j} \in \mathcal{Y}$ the jth output symbol ($1 \le j \le 4n'_\varepsilon$) and by $r_{l,i} \in \{0,1\}$ the ith output bit ($1 \le i \le 2n'_\varepsilon$) received by B in the lth iteration of $\widehat{\mathrm{OT}}$.

Let δ be as follows:

$$\delta = \min_{\substack{x\in\mathcal{X},\, y\in\mathcal{Y},\\ \mu_x:\, \mu_{x_1}=0 \vee \mu_{x_2}=0}} \left| \frac{P_{Y|X=x_1}(y) + P_{Y|X=x_2}(y)}{2} - \sum_x \mu_x P_{Y|X=x}(y) \right| ,$$

for some $\mu_x \in [0,1]$, $\sum\limits_x \mu_x = 1$. Let $\hat{y}$ be the output symbol for which the above minimum is achieved. Roughly speaking, δ is the best precision with which the cheating sender can simulate "the middle point" between the distributions $P_{Y|X=x_1}$ and $P_{Y|X=x_2}$ using the forbidden symbols.

Let κ be as follows:

$$\kappa = \min_{x\in\{x_1,x_2\}} \left| p_g - P_{Y|X=x}(y_1) P_{Y|X=x}(y_2) \right| .$$

Protocol 3.3 $\widehat{\mathrm{OT}} \to \mathrm{OT}$

1. A picks n_ε random bits $b_{1,0}, b_{2,0}, \ldots b_{n_\varepsilon,0}$ and sets $b_{l,1} = b_0 \oplus b_1 \oplus b_{l,0}$ for $1 \le l \le n_\varepsilon$.
2. B picks n_ε random bits $c_1, c_2, \ldots, c_{n_\varepsilon}$.
3. **Repeat** for $l = 1, \ldots, n_\varepsilon$
   (a) A runs $\widehat{\mathrm{OT}}(b_{l,0}, b_{l,1})(c_l)$ with B who gets b'_l,
   (b) **if** $d_H(r_{l,I_{l,c_l}}, r'_{l,I_{l,c_l}}) > \psi\varphi n'$ **then** B aborts.
4. **if** $\left|\#\{(l,j) | q_{l,j} = \hat{y}\} - 2n'_\varepsilon\big(P_{Y|X=x_1}(\hat{y}) + P_{Y|X=x_1}(\hat{y})\big)\right| > \delta n_\varepsilon/4$,

   then B aborts **else if**

   $$\left|\#\{(l,i) \mid r_{l,i} = y_1y_2 \text{ or } r_{l,i} = y_2y_1\} - 2p_g n'_\varepsilon\right| > \kappa n_\varepsilon/4 \ , \tag{3.7}$$

   then B aborts **else** B computes and sends $c' = c \oplus \left(\bigoplus\limits_{l=1}^{n_\varepsilon} c_l\right)$.
5. A computes and sends $\hat{b}_0 = b_0 \oplus \left(\bigoplus\limits_{l=1}^{n_\varepsilon} b_{l,c'}\right)$ and $\hat{b}_1 = b_1 \oplus \left(\bigoplus\limits_{l=1}^{n_\varepsilon} b_{l,1-c'}\right)$ to B.
6. B computes and outputs $\hat{b}_c \oplus \left(\bigoplus\limits_{l=1}^{n_\varepsilon} b'_l\right)$.

Theorem 3.2 *Protocol 3.3 implements OT with failure probabilities negligible in n.*

Before proving this theorem, let us introduce some other useful results. The next Lemma follows from Theorem 1 of [CK88]:

Lemma 3.4 *Suppose that Protocol 3.2 is executed n_ε times with parameters chosen according to Equations (3.4) and (3.5), and the corrupt A has got an advantage in guessing an auxiliary selection bit c_i (compared to a random guessing) in each execution independently with probability $\hat{p} < 1$ constant in n. Then in the resulting protocol $OT(b_0, b_1)(c)$, A's advantage in guessing c will be negligible in n.*

The test of Step 3 is to decide whether the syndrome sent by A was valid: if the decoded word is at Hamming distance larger than $\psi\varphi n'$ from the received one, then the syndrome was not correct. The following observation was made by Damgård: in fact, the cheating A may send a "slightly" incorrect syndrome but this does not give her any advantage because as long as B successfully corrects the errors, the players appear to agree on a well-defined word which looks completely random for B, exactly as required by the protocol.

More formally, let us denote $r_{l,I_{l,c_l}}$ as r and $r'_{l,I_{l,c_l}}$ as $r' = r \oplus e$ where e is an error vector. Suppose that the cheating A announces a syndrome $\tilde{s} = syn(\tilde{r})$ which is a syndrome of some word $\tilde{r} \neq r$ that can be written as $\tilde{r} = r \oplus \tilde{e}$ where $\tilde{e}$ is a distortion introduced by A. According to the discussion in Section 2.4, B calculates $syn(r') \oplus syn(\tilde{r}) = syn(e \oplus \tilde{e})$, uses the decoding algorithm to obtain the coset leader $\hat{e}$ and finally decodes $r'' = r' \oplus \hat{e}$. Now, if $w_H(\tilde{e})$ is small enough so that $w_H(e \oplus \tilde{e}) \leq \psi\varphi n'$ then the errors are corrected: $\hat{e} = e \oplus \tilde{e}$, and B accepts $r'' = r \oplus e \oplus e \oplus \tilde{e} = r \oplus \tilde{e}$. It follows that the cheating A could have just behaved honestly and sent $r \oplus \tilde{e}$ instead of r because in this case, her input and B's output would still be well-defined and consistent with the distributions prescribed in the protocol. One can use a standard simulation argument to show that more formally. Concluding, if A adds a slight distortion $\tilde{e}$, she will most likely pass the test of Step 3 but this will be equivalent to the honest behaviour. Hence, A must render the syndrome "completely incorrect" in order to perform her attack successfully, and the test of Step 3 is designed to prevent it. The argument above implies:

Lemma 3.5 *Suppose that in Step 3(a) of Protocol 3.3, the corrupt A sends to the honest B an incorrect syndrome $syn(r_{l,I_{l,c_l}} \oplus \tilde{e})$ then either B accepts it hereby agreeing on a string $r_{l,I_{l,c_l}} \oplus \tilde{e}$ with A, or B rejects it.*

B reads the auxiliary inputs $b_{l,0}$, $b_{l,1}$ at random, therefore if A renders one of the syndromes incorrect then she will fail the test of Step 3(a) with probability $1/2$ in each iteration. Hence, A fails at least one of the n_ε tests except with probability negligible in n.

We now argue that the tests of Step 4 achieve their goals. We use Lemma 2.1 which basically says that in n Poisson trials (where n is large enough), the probability that the number of experiments where an event in question occur is different from the expectation on some value νn is exponentially small in n and ξ^2. Observe that even if ν depends on n, the aforementioned probability will still be exponentially small in n as long as ν is asymptotically smaller than $1/\sqrt{n}$), or, equivalently νn is asymptotically bigger than $\sqrt{n}$.

In our setting, B shall count a number of good pairs and a number of individual symbols received. The transition matrix of the DMC is known to

B, so he can calculate expectations for the received pairs and symbols which correspond to A' honest behaviour. Note that in Protocol 3.3, the cheating A must send n_ε wrong pairs or forbidden symbols in order to gain any bias in guessing c. No matter her cheating strategy (i.e., what fraction of the wrong pairs/forbidden symbols to send), she will send at least $n_\varepsilon/2$ wrong pairs *or* at least $n_\varepsilon/2$ forbidden symbols. In any case, she sends a number of pairs/symbols which is *linear* in n_ε and those pairs/symbols have the different distribution compared to what B expects to see in the honest case. We use Lemma 2.1 observing that in our scenario, νn corresponds to n_ε. Note that $n_\varepsilon = [n^{1+\varepsilon}] > \sqrt{n'_\varepsilon} = \sqrt{n \cdot n_\varepsilon}$, therefore the dishonest A will be caught by the statistical tests with high probability. It only remains to argue that the thresholds in the tests of Step 4 are set correctly.

Let

$$z_{i,j} = \begin{cases} 0 & \text{if } q_{i,j} \neq \hat{y} \\ 1 & \text{if } q_{i,j} = \hat{y}. \end{cases}$$

Lemma 3.6 *There exists a constant $\rho_1 < 1$ such that when A does not send the forbidden symbols then*

$$P\left[\left|\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{4n} z_{i,j} - 2n'_\varepsilon\big(P_{Y|X=x_1}(\hat{y}) + P_{Y|X=x_2}(\hat{y})\big)\right| > \delta n_\varepsilon/4\right] < \rho_1^n \tag{3.8}$$

holds, whereas, if she sends at least $n_\varepsilon/2$ forbidden symbols, then

$$P\left[\left|\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{4n} z_{i,j} - 2n'_\varepsilon\big(P_{Y|X=x_1}(\hat{y}) + P_{Y|X=x_2}(\hat{y})\big)\right| < \delta n_\varepsilon/4\right] < \rho_1^n \tag{3.9}$$

holds.

Proof. (Sketch). When A follows the protocol, we have

$$E\left(\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{4n} z_{i,j}\right) = 4n'_\varepsilon \frac{P_{Y|X=x_1}(\hat{y}) + P_{Y|X=x_2}(\hat{y})}{2}, \tag{3.10}$$

This means that B expects to see the "middle distribution" between $P_{Y|X=x_1}$ and $P_{Y|X=x_2}$, in particular, for $\hat{y}$. Clearly, the best way for the cheating A (if she uses the forbidden symbols at all) to pass the test is to simulate the output distribution which is as close to the "middle distribution" as possible. However the choice of δ makes sure that when A cheats with at least $n_\varepsilon/2$ forbidden symbols, then:

$$\left|E\left(\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{4n} z_{i,j}\right) - 2n'_\varepsilon\big(P_{Y|X=x_1}(\hat{y}) + P_{Y|X=x_2}(\hat{y})\big)\right| \geq \delta n_\varepsilon/2$$

holds.

When setting the threshold $\delta n_\varepsilon/4$, Equation (3.8) follows immediately from Lemma 2.1 and Equation (3.9) follows from the same lemma by taking into

account (3.10) and observing that in this case the probability in (3.9) is in fact the tail probability.

Hereby, we make sure that the honest A is accepted with high probability while the cheater is rejected almost certainly. □

Let

$$w_{i,j} = \begin{cases} 1 & \text{if } r'_{i,j} = y_1y_2 \text{ or } r'_{i,j} = y_2y_1, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 3.7 *There exists a constant $\rho_2 < 1$ such that when A does not send incorrect pairs, then we have*

$$P\left[\left|\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{2n} w_{i,j} - 2p_g n'_\varepsilon\right| > \kappa n_\varepsilon/4\right] < \rho_2^n \ ,$$

whereas, if A sends at least $n_\varepsilon/2$ incorrect pairs,

$$P\left[\left|\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{2n} w_{i,j} - 2p_g n'_\varepsilon\right| < \kappa n_\varepsilon/4\right] < \rho_2^n.$$

Proof. (Sketch). For the second test of Step 4 the idea is that the receiver calculates the overall number of good pairs y_1y_2 and y_2y_1 on the output. If A follows the protocol, then we have the following expectation for this number:

$$E\left(\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{2n} w_{i,j}\right) = 2p_g n'_\varepsilon$$

where p_g is, as above, the probability to receive a good pair given that x_1x_2 or x_1x_2 was sent. At the same time, it A uses at least $n_\varepsilon/2$ incorrect pairs (i.e., x_1x_1 or x_2x_2) then by the choice of κ:

$$\left|E\left(\sum_{i=1}^{n_\varepsilon}\sum_{j=1}^{2n} w_{i,j}\right) - 2p_g n'_\varepsilon\right| \geq \kappa n_\varepsilon/2.$$

Setting the threshold $\kappa n_\varepsilon/4$ makes sure that the honest A is accepted with high probability while the cheater is rejected almost certainly according to Lemma 2.1. □

Now, we are ready to give the proof of Theorem 3.2.

Proof. The completeness property follows since Lemmas 3.3, 3.6 and 3.7 guarantee that the honest A is accepted almost always in the tests of Steps 3 and 4, respectively.

B's security for the case of passively cheating A is argued in Proposition 3.1. Assume that A behaves consistently with some inputs (b_0, b_1) (note that we put some requirements on B's output for this case only[5]). Now, if A tries to cheat

[5]See the discussion in Section 3.5.1

actively by sending wrong pairs or wrong syndromes, then Lemmas 3.6– and 3.5, respectively, make sure that she will be rejected with high probability.

It is only left to show that in Step 6, B indeed recovers b_c with high probability given that he accepts all the tests.

$$\hat{b}_c \oplus \left(\bigoplus_{l=1}^{n_\varepsilon} b'_l \right) \tag{3.11}$$

$$= b_c \oplus \left(\bigoplus_{l=1}^{n_\varepsilon} b_{l,c' \oplus c} \right) \oplus \left(\bigoplus_{l=1}^{n_\varepsilon} b_{l,c_l} \right) \tag{3.12}$$

$$= b_c \oplus \left(\bigoplus_{l=1}^{n_\varepsilon} b_{l, \bigoplus_{l=1}^{n_\varepsilon} c_l} \right) \oplus \left(\bigoplus_{l=1}^{n_\varepsilon} b_{l,c_l} \right) \tag{3.13}$$

$$= b_c \oplus b_{0,z} \oplus b_{0,z} \tag{3.14}$$

$$= b_c \tag{3.15}$$

Here, Equation (3.12) follows by expanding the expression for $\hat{b}_c$ (see Step 5) and the fact that $b'_l = b_{l,c_l}$ with high probability once the tests are accepted. The choice of c' in Step 4 implies (3.13) and then, (3.5), (3.6) imply (3.14).

Security for B follows from Proposition 3.1 for the case of passively cheating A, the equations above make sure that B obtains the correct bit, while Lemmas 3.6 and Lemmas 3.7 extend B's security for the case of actively cheating A.

Security for A is shown in Proposition 3.1 for one execution of Protocol 3.2. It is worth noting that the corrupt B may try to collect as much information as possible on A's inputs b_0 and b_1 trying to figure out the auxiliary inputs $b_{l,0}$ and $b_{l,1}$ in each iteration. However, Protocol 3.2 is repeated only a number of times polynomial in n and therefore, it is clear that the resulting B's information on at least one b_0 and b_1 will still be negligible in n.

This concludes the analysis of the protocol, and, hence, the proof of the theorem. □

3.4.4 String Oblivious Transfer

Note that *string* OT instead of *bit* OT could be obtained using hashing to a bit-string (rather than a bit) as the privacy amplification function. However, secret strings of large length l demand high communication complexity. Despite of the fact [KM01] that in the asymptotic case, for any l which is fixed there always exists n, so that OT based on any non-trivial BSC is secure, the generic reductions from Bit to String OT – introduced in [BCR86] and later elaborated in [BCS96, BC97, BCW03] – may be more efficient for some applications.

3.4.5 Special Case: Binary Symmetric Channel

We shall consider next various issues concerning asymptotic communication complexity of the presented OT protocol for the special case when the DMC

connecting the players is a δ-BSC.

Original Protocol

In this section, we consider the original OT protocol of [Cré97] with a modification for transferring bit-strings instead of bits. We shall not recall the whole construction but only describe the corresponding modifications in the protocols of Section 3.4.

The first test of Step 4 is removed from Protocol 3.3. This can be safely done since the test ensures that A can only send the two prescribed input symbols that is obviously redundant in the binary case.

Consequently, we have $\mathcal{X} = \mathcal{Y} = \{0,1\}$ and w.l.o.g. we assume that $x_1 = y_1 = 0$, $x_2 = y_2 = 1$ and $P_{Y|X=x_1}(y_2) = P_{Y|X=x_2}(y_1) = \delta$. Then according to (3.2):

$$\varphi = \frac{\delta^2}{\delta^2 + (1-\delta)^2} \tag{3.16}$$

Note that B can always fill one of the subsets with good pairs since according to (3.3)

$$p_g = \delta^2 + (1-\delta)^2, \tag{3.17}$$

that is bigger than $1/2$ once $0 < \delta < 1/2$, therefore $n' = n$. On the other hand, it is clear that all the bad pairs are erasures from B's point of view, i.e., they contain no information about the corresponding input bit.

In Protocol 3.1, A will now encode her random input using the two-repetition code. Note that the security proof of Subsections 3.4.1–3.4.3 is still valid in this case, no matter how A encodes her random input bit x: $x \to (x,x)$ or $x \to (x, 1-x)$, both are kinds of the two-repetition code. However, the encoding $x \to (x,x)$ simplifies the presentation of this subsection.

Finally, in order to transmit the bit-strings b_0, b_1 of length l, A and B agree on a random 2-universal hash function g: $\{0,1\}^n \to \{0,1\}^l$, and then, in Protocol 3.2, A encrypts her inputs as follows $\hat{b}_i = b_i \oplus g(r_{I_i})$ for $i = \{0,1\}$ where "$\oplus$" is a bitwise XOR. Consequently, Step 5 of Protocol 3.2 and Protocol 3.3 have to be changed in a corresponding way. It is easy to verify that the protocols generate correct output in the case when A's inputs are the bit-strings and the operation "$\oplus$" is bitwise XOR.

Remark 3.3 The second test of Step 4 can be sufficiently simplified observing that in binary case there are output pairs of only two possible qualities: the informative ones and the erasures "Δ". Therefore, the Equation (3.7) can be replaced with

$$\#\{(l,i) \mid r_{l,i} = y_1 y_2 \text{ or } r_{l,i} = y_2 y_1\} < 2p_g n'_\varepsilon - (p_g - 1/2) n_\varepsilon$$

as it follows from Theorem 4 of [Cré97]. We sketch the argument (which is analogous to the ones in Lemmas 3.6, 3.7) here. Let $w_{i,j} = \begin{cases} 1 & \text{if } r'_{i,j} \neq \Delta \\ 0 & \text{if } r'_{i,j} = \Delta \end{cases}$.

If A behaves honestly, then clearly we have $E\left(\sum_{i=1}^{n_\varepsilon}\sum_{i=1}^{n} w_{i,j}\right) = 2p_g n'_\varepsilon$. At the same time, if A sends at least n_ε wrong pairs, then we get

$$E\left(\sum_{i=1}^{n_\varepsilon}\sum_{i=1}^{n} w_{i,j}\right) \leq p_g(2n'_\varepsilon - n_\varepsilon) + (1-p_g)n_\varepsilon = 2p_g n'_\varepsilon - (2p_g - 1)n_\varepsilon.$$

Then, by setting the threshold $(2p_g - 1)n_\varepsilon/2 = (p_g - 1/2)n_\varepsilon$, we ensure that the honest A is accepted while the cheating A is rejected with high probability according to Lemma 2.1.

Security Analysis

It was claimed in [Cré97] that the Passive OT Protocol 3.2 (and, consequently, Protocol 3.3) could be proved secure for $\delta < 0.1982$ while the security argument for the noisier channel was left as an open question.

However, the very same protocol appears to build a secure OT for any δ, $0 < \delta < 1/2$, as it was realised in [KM01]. We shall sketch this argument here because it also gives some insights in the complexity of Protocol 3.2 which immediately carries over to the complete Protocol 3.3 since the complexity of the latter polynomially bounded by that of the former. We shall see that there exist certain values of the error rate δ which are optimal from the communication complexity point of view.

Recall the security argument for Protocol 3.2. The best strategy for the dishonest B is to split the accepted bits evenly over the two sets. The protocol remains secure as long as the syndromes provided by the honest A do not contain enough information for correcting all the errors in both sets. Now, we are in the scenario of the privacy amplification theorem (Theorem 2.3) where $R(W|V=v)$ is B's entropy of one set,[6] l is the length of bit-strings b_0, b_1 and r is the length of the syndromes. Since we want B's information on at least one of b_0, b_1 be negligible in n, we require:

$$R(W|V=v) - l - r > 0. \tag{3.18}$$

The connection to the communication complexity of the protocol is the following: the larger the right part of (3.18) is, the smaller code lengths n are needed in order to provide the required (fixed) failure probabilities.

It only remains to express all the terms in (3.18) as functions of δ. According to Lemma 2.2, we substitute $R(W|V=v)$ with corresponding Shannon information since we are in asymptotic case. The important observation to make here (which was missing in [Cré97]) is that only accepted bits contain any information while erasures contain none. Thus, we write $R(W|V=v) = (1-p_g)n + p_g h(\varphi)n$. The number of check bits r can be taken to be $nh(\varphi)$ according to Theorem 2.2.

We take $l = \alpha n$ and we call α the *rate* of the protocol, the notion which quite naturally arises here. A thorough analysis of the OT rate can be found in [WN04].

[6] In fact, we do not have to specify which one, since we demand that *at least one* of the bits must be lost by B.

Plugging all the terms into (3.18) and dividing by n, we have

$$f(\delta) = (1 - p_g) + p_g h(\varphi) - \alpha - h(\varphi)$$
$$= 2\delta(1-\delta)\left(1 - h\left(\frac{\delta^2}{\delta^2 + (1-\delta^2)}\right)\right) - \alpha. \quad (3.19)$$

This function for the case of $\alpha = 0$ is depicted on Figure 3.1. We stress that since we consider the asymptotic case (i.e., $n \to \infty$), $\alpha = 0$ means that l is a constant in n.

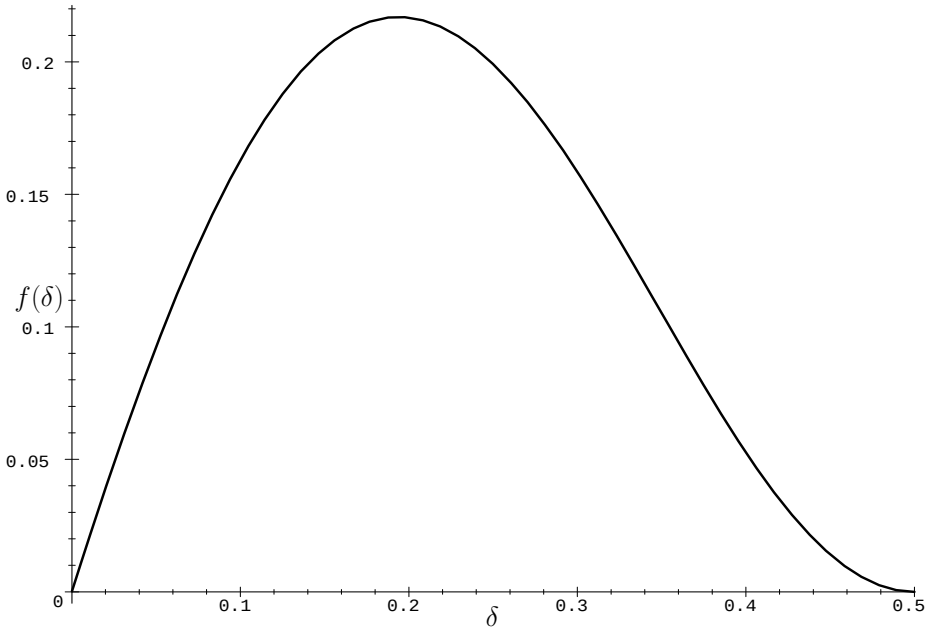

Figure 3.1: Exponent $f(\delta)$ for $\alpha = 0$

Figure 3.1 shows that in asymptotic case, OT is always achievable from any non-trivial δ-BSC, $0 < \delta < 1/2$, even if the strings of constant (in n) length are transferred. On the hand, it is clear from (3.19) that whenever $\alpha > 0$, i.e., l constitutes a fraction of n, the range of achievable error rates gets restricted from both below and above. In the extreme case of $\alpha > 0.217$, no secure String OT can be constructed this way. We stress that α is the rate of Passive OT Protocol 3.2 while the rate of the Complete Protocol 3.3 is clearly zero since the complexity of the latter is $O(n^{2+\epsilon})$, $\epsilon > 0$.

Protocol with Multiple Repetition Code

It is very natural to generalise Protocol 3.1 by having A encode her random input using an a-repetition code. B accepts a received a-bit string if and only if the number of zeroes or ones in this string is at least b, $b < a$.

This results in the following expression for the probability that B accepts a

symbol:

$$p'_g = \sum_{i \in [0,a-b] \cup [b,a]} \binom{a}{i} \delta^i (1-\delta)^{a-i}, \tag{3.20}$$

while the error rate for the accepted symbols is

$$\varphi' = \frac{1}{p'_g} \sum_{i=b}^{a} \binom{a}{i} \delta^i (1-\delta)^{a-i}. \tag{3.21}$$

Now, we can perform an analysis which is analogous to the one in the previous subsection. We take $r = nh(\varphi')$ as above. We have the following expression for the average collision entropy of each symbol received by B: taking the average is necessary since the entropy depends on the Hamming weight of the received symbol:

$$R(W|V=v) = \frac{1}{2} \sum_{i=0}^{a} \binom{a}{i} \left(\delta^i (1-\delta)^{a-i} + \delta^{a-i} (1-\delta)^i \right) \cdot h\left(\frac{\delta^i (1-\delta)^{a-i}}{\delta^i (1-\delta)^{a-i} + \delta^{a-i} (1-\delta)^i} \right). \tag{3.22}$$

One can now substitute the expressions (3.20–3.22) into (3.18) and perform the asymptotic analysis for the whole family of the OT protocols which are modified for the use of (a, b)-repetition codes described above.

Example 3.1 Numerical calculations show that the function $f(\delta)$ of (3.19) reaches the maximum value of 0.21 for $\delta = 0.2$, so this error rate appears to be optimal with respect to the communication complexity of the protocol using two-repetition code. At the same time, the similar function for the scheme with $(a = 8, b = 6)$-repetition reaches the maximum value of 0.285 for $\delta = 0.327$. This is a slightly better result (compared to the previous one) which is achieved for the higher error rate.

The further generalisation for the use of general linear codes instead of repetition codes was considered in [KM01]. This research is out of the scope of this thesis.

3.5 Concluding Remarks

3.5.1 Input Awareness

As it was mentioned in Section 3.2.3, our protocols do not satisfy the input awareness property, since for instance, in Protocol 3.2, the cheating B may input "?" by splitting the good bits evenly over the two sets. Strictly speaking, his selection bit c is undefined in this case. As for the cheating A, she may misbehave in Protocol 3.3 by choosing the auxiliary inputs $b_{l,0}$ and $b_{l,1}$ for $l = 1, \ldots, n_\varepsilon$ completely at random and otherwise behave honestly. It is easy to

see that it is impossible to define input bits b_0 and b_1 in this case. Consequently, the security for B is not in place because one cannot argue that B receives b_c for $c = \{0, 1\}$.

The way to gain the input awareness property is to have the players commit to their inputs and prove in zero-knowledge that they behaved correctly. This construction was introduced in [Cré89] as *Verifiable Oblivious Transfer* .

3.5.2 Special Cases

Binary Channel

In the special case when the DMC is a binary channel, a simple modification to the protocol of [Cré97, SW02] is enough in order to obtain secure OT from such a channel. The idea is to have the sender A encode her random input as a pair of different bits rather than using a repetition code, i.e., to encode "0" as "01" and "1" as "10", for instance.

In this case, it is clear that the output yy, $y \in \{0, 1\}$ is an erasure from the receiver's point of view, while for A, the channel after encoding looks like a BSC. Then exactly the same argument as in [Cré97, SW02] yields the secure OT from any binary channel.

Error-free Symbols

Note that the probability φ might be equal to 0. It is the case when y_1 or y_2 are not in the joint support of x_1 and x_2, i.e., $P_{Y|X=x_1}(y_2) = 0$ or $P_{Y|X=x_2}(y_1) = 0$, respectively. This implies that receiving such symbols is equivalent to receiving x_1 or x_2 in clear. In this case, the decoding algorithm in Protocol 3.1 has to be modified in the following way:[7]

$$B \text{ outputs } \begin{cases} 0 & \text{if } y_1 y_{1,2} \text{ or } y_{1,2} y_2 \text{ is received} \\ & \quad \text{and } \big(P_{Y|X=x_1}(y_2) = 0 \text{ or } P_{Y|X=x_2}(y_1) = 0\big) \\ 1 & \text{if } y_{1,2} y_1 \text{ or } y_2 y_{1,2} \text{ is received} \\ & \quad \text{and } \big(P_{Y|X=x_1}(y_2) = 0 \text{ or } P_{Y|X=x_2}(y_1) = 0\big) \\ \Delta & \text{if any other pair is received.} \end{cases}$$

Then the error correction in Protocol 3.2 is not needed. However, all the checks of Step 4 in Protocol 3.3 have to be performed to prevent A's active cheating.

Rabin Oblivious Transfer

In the special case when there exists only one $y \in \mathcal{Y}$ such that $P_{Y|X=x_1}(y) \neq 0$ and $P_{Y|X=x_2}(y) \neq 0$, i.e., the only one y is in the joint support of x_1 and x_2, it is easy to see that Protocol 3.2 implements a Rabin OT. Hence, the standard technique of [Cré87] can be used to turn it into OT. Nevertheless, the test introduced in Step 4 of Protocol 3.3 has still to be performed.

[7] Here, the expression "$y_1 y_{1,2}$" denotes the case when the first received symbol in the pair is y_1 while the second one may be either y_1 or y_2.

3.5.3 Open Questions

Oblivious Transfer with Non-zero Rate

In [WNI03], a rate of unconditionally secure primitives is introduced and later in [WN04], it is (quite naturally and somewhat analogously to [KM01]) defined as the ratio of the secrets' length to the number of channel uses. As we mentioned above, the rate of our Protocol 3.3 is zero.

The open question (also posed in [WN04]) is to realise String OT (secure against active cheating) with non-zero rate based on noisy channel.

Oblivious Transfer from More General Channels

Another objective is to realize OT from more general and practically important channels, such as *continuous alphabet* channels or channels *with memory*.

Chapter 4

Binary Symmetric Channel: Practicality Issues

4.1 Introduction

This chapter is focused on *non-asymptotic analysis* of OT and BC primitives based on Binary Symmetric Channel introduced by Crépeau [Cré97]. These primitives are proved to be secure in asymptotic case [Cré97, KM01, SW02], i.e., when the number of the noisy channel's uses is large enough. At the same time, when given an implementation of BSC, it is important to find out what this "large enough" means for some fixed values of the failure probabilities (i.e., the security requirements) – that is a purpose of non-asymptotic analysis.

We investigate here only the primitives based on Binary Symmetric Channel. We admit that this model is not entirely practical meaning that it quite roughly describes the real communication channels. However, the material of this chapter is to be seen as the first insight into non-asymptotic behaviour of the unconditional primitives based in noisy channels with an attempt to establish the trade-offs between their communication complexity on the one hand and their security on the other hand.

This chapter is based on [KM01, KM00].

4.2 Preliminaries

4.2.1 Communication Model

We assume that the sender A and the receiver B are connected by a Binary Symmetric Channel with an error rate δ $(0 < \delta < 1/2)$ as well as a noiseless channel.

4.2.2 On Error-Correcting Codes Used

In Chapter 3, we exploited concatenated codes due to their good asymptotic performance. On the contrary, we are now going to deal with the codes of a finite length. For the sake of simplicity of our analysis, we shall use BCH codes. It is well-known that their asymptotic performance is not good (see,

e.g. [MS77], Ch. 9. § 5), however for the lengths up to a few thousands, these codes are among the best known ([MS77], Ch. 9. § 4), hence they will suite our needs.

Another advantage of using BCH codes is that it is easy to estimate the minimal code distance d of some particular $[n, k, d]$-code. According to [PW72], Theorem 9.2, for any positive integer m and $t_0 < n/2$ there exists a binary BCH code C of length $n = 2^m - 1$ correcting t_0 (or less) errors and containing no more than mt_0 check bits.

The next corollary follows taking into account that the following expressions hold for the so called *designed distance* d_0 [PW72]:

$$d_0 = 2t_0 - 1 \tag{4.1}$$

and

$$d > d_0. \tag{4.2}$$

Corollary 4.1 *For a BCH $[n, k, d]$ code with $n = 2^m - 1$ where m is a positive integer, the number of check bits r can be estimated as follows*

$$r \leq m(d-1)/2,$$

assuming $d = d_0$.

4.3 Oblivious Transfer

For the simplicity sake, we will discuss in this section only the case of δ-BSC, where $0 < \delta < 1/2$. However, according to the discussion of Subsection 3.5.2, the results given here carry over straightforwardly for the case of any binary channel.

We shall analyse the modified Protocol 3.3 as discussed in Subsection 3.4.5. Whenever we refer to Protocol 3.3 in this chapter, we mean the modified one.

Recall that for the case of δ-BSC, the expressions for error rate of the good pairs φ and the probability for B to receive a good pair p_g are given in (3.16) and (3.17), respectively.

4.3.1 Non-Asymptotic Analysis

We shall define failure probabilities for Protocol 3.3, then derive their dependencies on the size n of the used BCH code C which in turn characterises a communication complexity of the protocol. Finally, we give some numerical calculation results.

The failure probabilities are defined as follows:

- P_c is the probability that completeness fails, i.e., the honest A is rejected by the honest B.
- P_A, A's security failure probability that is the dishonest B's advantage in guessing the secret bit b_{1-c} compared to a random guess.

- P'_B, P''_B, B's security failure probabilities which are, respectively, the dishonest A's probability to gain *any* non-zero advantage in guessing the selection bit c compared to a random guess and the probability that B obtains $1 - b_c$, i.e., the incorrect choice bit.

Remark 4.1 On input awareness: We assume that A behaves consistently with some input bits b_0, b_1 therefore the choice bit b_c is well-defined for the honest B. The dishonest B may input "?" by splitting good pairs over the two sets, then his choice bit c is indeed undefined. However, we observe that B might receive one of the inputs just by cheating passively. On the other hand, he cannot do any worse than splitting the good pairs *evenly* over the two sets. Then, we can conservatively assume that B gets one of the bits (the one which he has most information about) for free. Hence, we can make our analysis for the set which is filled with half of the good pairs. If the dishonest B has only a negligible information about the corresponding A's input (and we think of it as b_{1-c}), then we ensure that A's security holds.

Let us consider Protocol 3.3 and first derive the relation for the completeness failure probability. We denote n_ε by N.

Remark 4.2 In our analysis, we shall assume δ to be significantly smaller than $1/2$. This is justified by the analysis of Subsection 3.4.5 where the optimal δ is shown to be equal to 0.2. Recall that in the previous chapter, we used the asymptotically good codes which correct errors close to the channel capacity. On the contrary, the BCH codes which we use here have significantly worse error-correction capability, therefore we expect the optimal δ to be less than 0.2. The assumption helps to simplify our analysis, but nevertheless, we give a precise relation for each failure probability as well.

Note that the fact that δ is significantly smaller than $1/2$ implies that ε is significantly bigger than $1/2$ (denoted $\varepsilon >> 1/2$). In the statement of the following lemma, saying "n large enough", we mean "large enough such that the Chernoff bounds (2.1), (2.2) gives us a good estimate". In our analysis, we shall need n of about a few thousands, so the Chernoff bound will indeed provide us with a good estimate.

Lemma 4.1 *Completeness failure probability P_c can be expressed for large enough n and $\varepsilon >> 1/2$ as follows:*

$$P_c = \max(P_{c1}, P_{c2}), \text{ where}$$

$$P_{c1} < \exp\left(-\frac{N(p_g - 1/2)^2}{4p_g n}\right), \text{ and} \tag{4.3}$$

$$P_{c2} < 1 - \left(1 - \tilde{P}_{c2}\right)^N, \text{ where} \tag{4.4}$$

$$\tilde{P}_{c2} < \left(\frac{\exp(\xi)}{(1+\xi)^{(1+\xi)}}\right)^{n\varphi}, \ \xi = \left\lfloor \frac{d+1}{2} \right\rfloor \frac{1}{n\varphi} - 1.$$

Proof. There are two possibilities for the honest A to be rejected by the honest B: the second test of Step 4 and the test of Step 3, both in Protocol 3.3. We shall estimate the honest A's failure probability for both tests and choose P_c to be the maximum of the two.

For the former test, the honest A fails it, if too few good bits were received by B, namely less than $2p_gnN-(p_g-1/2)N$ good bits according to Remark 3.3. We remind that $N = n_\varepsilon$ and so $Nn = n'_\varepsilon$. It easy to see that the probability of A's failure is upper bounded by the following sum:

$$P_{c1} < \sum_{i=0}^{2p_gnN-(p_g-1/2)N} \binom{2nN}{i} p_g^i (1-p_g)^{2nN-i},$$

which can be estimated using the Chernoff bound (2.2) yielding Equation (4.3).

Let us consider now the test of Step 3, Protocol 3.3. Clearly, the honest A fails it whenever more than $\psi\varphi n$ errors occur during the transmission of the set r_{I_c} in at least one iteration of the auxiliary Protocol 3.2, while the error rate for each pair in r_{I_c} is φ. The probability that too many errors occur in r_{I_c} is estimated as follows

$$\tilde{P}_{c2} < \sum_{i=\psi\varphi n}^{n} \binom{n}{i} \varphi^i (1-\varphi)^{n-i}$$

for each execution of Protocol 3.2. The value $\psi\varphi n$ is the number of errors which the code is capable to correct. Taking into account Equations (4.1) and (4.1), we assume $\psi\varphi n = \left\lfloor \frac{d+1}{2} \right\rfloor$ where d is the minimal code distance of the chosen code C. The Chernoff bound (2.1) implies the estimate (4.4) for $\tilde{P}_{c2}$. Finally, Equation (4.4) has to hold in all N (independent) executions of Protocol 3.2 hence $P_{c2} < 1 - \left(1 - \tilde{P}_{c2}\right)^N$. □

We consider next the failure probability for A. Let n and r be the length and the number of check bits of the chosen code C, respectively.

Lemma 4.2 *If B is cheating, then his bias P_A in guessing b_{1-c} (compared to a random guess) is expressed as follows:*

$$P_A = \tilde{P}_A - 1/2,$$

where $\tilde{P}_A$ is chosen according to

$$1 + (1-\tilde{P}_A)\log(1-\tilde{P}_A) + \tilde{P}_A \log \tilde{P}_A < N \cdot I_r, \tag{4.5}$$

where

$$I_r < (1-P_m)2^{-(R-r-1)}/\ln 2 + P_m, \tag{4.6}$$

where

$$R = n - \gamma\left(1 + \log\left(\varphi^2 + (1-\varphi)^2\right)\right), \tag{4.7}$$

$$P_m \le \left(\frac{exp(\tilde{\xi})}{(1+\tilde{\xi})^{(1+\tilde{\xi})}}\right)^{2p_gn}, \text{ where } \tilde{\xi} = \frac{\gamma}{p_gn} - 1, \tag{4.8}$$

and γ satisfies

$$p_g n << \gamma < \frac{n-r-1}{1+\log\left(\varphi^2+(1-\varphi)^2\right)} \tag{4.9}$$

Proof. Let us first consider a single execution of Protocol 3.2. We perform our analysis under a conservative assumption that the dishonest B uses the best adversarial strategy which is, as shown in Subsection 3.4.2, the following: B distributes the good bits evenly over the two sets r'_{I_c} and $r'_{I_{1-c}}$ trying to get as much information as possible about both bits b_c and b_{1-c}. According to Remark 4.1, we think of b_{1-c} as the input bit corresponding to the set which the cheating B filled with a half of all received good pairs.

Fano's inequality (2.3) allows us to estimate the probability that B decodes b_{1-c} correctly when given his Shannon's information about b_{1-c}. Hence, the left part of (4.5) follows taking into account that b_0 and b_1 are the bits. We argue the right part of (4.5) next. We denote by I_r the information about b_{1-c} which the cheating B learns from one instance of Protocol 3.2. The difficulty in estimating this information is that I_r may vary from one execution of Protocol 3.2 to the other depending on the number of good bits which B receives in each execution. However, we can upper bound this information as in (4.6) by setting a certain threshold γ on the number of good bits which B can place in each set r'_{I_c} and $r'_{I_{1-c}}$. If this number is less than γ then we use the privacy amplification theorem (Theorem 2.3) to estimate I_r otherwise, we assume that the cheating B got too much information about b_{1-c} and we conceptually give away this bit to B. The rational behind this idea is that we expect a probability P_m that the threshold γ is exceeded to be very small. We shall discuss the choice of γ below. As for the exponent in (4.6), we note that in our scenario, R is B's Rényi entropy of $r'_{I_{1-c}}$, so (4.7) follows by the definition of Rényi entropy, its additivity and the following argument: Observe that only the good pairs carry some information while the bad pairs, i.e., erasures clearly do not. Summarising, we have $n-\gamma$ bits of entropy contributed by erasures while each accepted bit clearly contributes $-\log\left(\varphi^2+(1-\varphi)^2\right)$.

We argue (4.8) next. Observe that the probability P_m that the threshold γ is exceeded is in fact a probability that B receives more than 2γ good bits in one execution of Protocol 3.2. P_m can be expressed as follows:

$$P_m = \sum_{i=2\gamma}^{2n} \binom{2n}{i} p_g^i (1-p_g)^{2n-i},$$

while estimating it using the Chernoff bound (2.1) gives us (4.8).

Note that we want to upper bound the total amount of information which the dishonest B can collect about b_{1-c} in Protocol 3.3. Consider the way A forms the auxiliary inputs $b_{l,0}, b_{l,1}$ for $1 \geq l \geq n_\varepsilon$ in Step 1 of Protocol 3.3. Without loss of generality, we can assume that the dishonest B always gets $b_{l,0}$ and learns b_0 for free[1] while trying to guess $b_{l,1}$. In this case, I_r is an upper bound on the information which B obtains about $b_{l,1}$, however B learns $b_{l,0}$ and

[1]Recall that we took for granted from the beginning that the cheating B learns one of the bits.

eventually b_0. Hence, as a matter of fact, I_r is his information on b_1, i.e., b_{1-c}. It is clear now that B cannot learn more than $N \cdot I_r$ bits of information about b_{1-c} from N executions of Protocol 3.2, so the right part of (4.5) follows.

Finally, we argue the choice of γ as in (4.9). Clearly, we can estimate I_r as in (4.6) only when the privacy amplification theorem holds, i.e., when the exponent (4.6) is negative. This condition is expressed in the right part of (4.9). On the other hand, a half of the average number of good bits received by B (which is $p_g n$) must be significantly smaller than γ such that the Chernoff bound is applicable, this relation is denoted by "<<" in (4.9). This completes the proof. □

Lemma 4.3 *The probability for the dishonest A to obtain any non-zero advantage in guessing b_c is*

$$P'_B < \exp\left(-\mu\xi''^2/2\right), \tag{4.10}$$

where $\mu = 2p_gNn - (2p_g - 1)N'$, $\xi'' = \frac{2p_gNn-(p_g-1/2)N'}{E} - 1$,

and N' is the overall number of incorrect pairs sent by A.

Proof. There are two ways for the dishonest A to cheat in Protocol 3.3, assuming that she behaves consistently with some inputs b_0, b_1. A can send wrong syndromes in Step 3(a) but according to Lemma 3.5, if she is accepted then this is equivalent to her honest behaviour, otherwise she is rejected and B's security holds. We conclude that we only need to analyse A's attack with sending wrong pairs – the second possible way for her to gain a bias about B's selection bit.

As argued in the proof of Lemma 4.1, A is accepted by the honest B if he gets at least $2p_gNn - (p_g - 1/2)N$ good bits. At the same time, if A sends N wrong pairs then the expectation μ of the number of good bits in the output will be different from the threshold above. More precisely, we have:

$$\mu = p_g(2Nn - N') + (1 - p_g)N' = 2p_gNn - (2p_g - 1)N'.$$

And so, the Chernoff bound (2.2) gives us an estimate (4.10) for the probability that the actively cheating A is not rejected. As a matter of fact, μ is going to be significantly less than the threshold, because (as it was pointed out in the beginning of Section 3.4.3) A must send at least N wrong pairs in Protocol 3.3 – at least one wrong pair in each iteration of Protocol 3.2 – in order to get any non-zero advantage in guessing c.

□

Lemma 4.4 *$P''_B = P_{c2}$ for large enough n and $\varepsilon >> 1/2$.*

Proof. Note that once A behaves consistently with some inputs b_0, b_1 (and in our analysis, we do not impose any requirements otherwise), then the analysis of Section 3.4 implies that it is impossible for the cheating A to have B output the incorrect bit $1 - b_c$ unless B fails to correct all the errors in the good set r_{I_c} in

each iteration of Protocol 3.2. The lemma follows observing that Equation (4.4) gives an estimate for this probability. □

Using Lemmas 4.1–4.4, the following numerical results have been obtained.

Example 4.1 Fix $l = 1$ (i.e., let b_0,b_1 be the bits) and the required failure probabilities to be $P_c = P_A = P'_B = P''_B = 10^{-6}$.

Then assuming a δ-BSC with a favourable error rate $\delta = 0.045$, one has to choose a code C of length $n = 65535$ with $r = 4000$, $d = 501$. Protocol 3.2 has to be iterated $N = n^{1.513} \cong 2 \cdot 10^7$ times that gives its communication complexity of about 296 Gigabytes.

4.3.2 Concluding Remarks

The non-asymptotic analysis of OT based on BSC has shown that in order to achieve the reasonable failure probabilities of 10^{-6}, one needs to pay for it with high communication complexity of some tens of Gigabytes.

We conclude that efficiency improvements are needed in order to make the current implementation of OT based on BSC practical.

4.4 Bit Commitment

In this Section, we consider the protocol of [Cré97] for Bit Commitment with modification for committing to an l-bit string b (instead of only one bit as in the original protocol). We set the security requirements, briefly sketch the security in the asymptotic case and then present a non-asymptotic analysis.

4.4.1 Security Definition

We require our protocol to satisfy the following properties:

- Completeness: if A and B are both honest, they accept the protocol with high probability.
- Hiding: If A is honest then the commitment reveals to B nothing about b. Formally, $D_{b=z}(View_B) \simeq D_{b=z'}(View_B)$ for all $z \neq z'$, $z, z' \in \{0,1\}^l$, where $D_{z=\{0,1\}^l}(\cdot)$ is the distribution over all possible B's views and A's inputs.
- Binding: If B is honest then he should always accept with some value b which the honest A wishes to commit to. Furthermore, A is unable to "change her mind" by opening another value. Formally:
  $P\,[A \text{ opens } b \text{ when committed to } b']$ is negligible for all $b \neq b'$ such that $b \in \{0,1\}^l$, $b' \in (\,\{0,1\}^l \cup \{?\}\,)$.

Remark 4.3 Defining b' this way, we demand a special kind of input awareness property. The dishonest A may commit to an illegal input "?", i.e., the value that is undefined from her point of view, however she will not be able to open it as any legal value in this case.

4.4.2 Protocol

Initialisation phase: The parties agree on using a linear $[n, k, d]$ code C with $k = (1 - H(\delta) + \xi)n$, $\xi > 0$ and $d \geq \beta\delta n$, where β is chosen such that $0 < \beta < 1$, $h(\delta) - \xi > h(\beta\delta)$; and a hash function $g : \{0,1\}^n \to \{0,1\}^l$ randomly chosen from a 2-universal class.

Protocol 4.1 Commit

1. Alice picks at random a code word $c \in C$ and computes an l-bit string x such that[2]
$$b = x \oplus g(c), \tag{4.11}$$
2. A sends c over δ-BSC and announces x to B,
3. B stores x and c', where c' is the received noisy version of c.

Protocol 4.2 Open

1. A announces the code word c to B,
2. B computes b using (4.11) given x and c. B accepts b, if $c \in C$ and $d_H(c, c') \leq (\delta + \kappa)n$, $\kappa < \beta\delta(1/2 - \delta)/2$, otherwise he rejects.

4.4.3 Asymptotic Analysis: Sketch

The code C is chosen at random so that it satisfies the conditions given in the initialisation phase with high probability according to Theorem 2.1.

Intuitively, the set of codewords of C is used as a "sparse" set and the commitments are chosen from this set. The transmission through the δ-BSC "shifts" the commitment c approximately to the distance of δn in the Hamming sense. Then, c' becomes an evidence for the commitment c.

Note that the conditions on the code C which are set in the initialisation phase imply that the code is not capable to correct all the errors in c' using the decoding algorithm of C with high probability. Hence, the hiding property is achieved since the computationally unbounded B may find the codewords which are δn-close to c' (in Hamming sense), but he is unable to tell them apart because they are all equally likely from his point of view as the candidates for being the actual c.

Formally, let $\mathcal{C}$, $\mathcal{C}'$ and X be the random variables describing c, c' and x, respectively, from B's point of view. Let G be a random variable describing the choice of g. Let $0 < \xi' < \xi$. Note that before committing takes place, $\mathcal{C} \in_R C$ and $X \in_R \{0,1\}^l$ uniformly for B. We apply Theorem 2.3 where $R(W|V = v) = (h(\delta) - \xi')n$ due to Lemma 2.2; and $r = n - k = (h(\delta) - \xi)n$ obtaining the following:

Lemma 4.5 *For all sufficiently large n:*

$$I(\mathcal{C} \oplus g(X); syn(\mathcal{C}) = (0, 0, \ldots, 0), \mathcal{C}', X, G) \leq 2^{-((\xi - \xi')n - l)} / \ln 2.$$

[2] In a particular case when b is a bit, the string x can be removed from (4.11).

The lemma basically says that one can always choose the constants ξ and ξ' such that B will be left with negligible (in n) amount of information about b, i.e., the scheme is indeed hiding.

On the other hand, the scheme is also binding because, intuitively, A does not know "where" c was shifted to by the BSC. She is able to find a codeword which can be opened as some $\tilde{b} \neq b$, but it is unlikely that this word will be δn-close to c' in the Hamming sense.

As a matter of fact, the dishonest A may send any word $w \in \{0,1\}^n$ in order to later claim it as some codeword which has been sent. We observe that no matter which w the cheating A sends, for any codeword $\tilde{c} \in C$ (excluding may be a certain $\bar{c}$ which satisfies $d_H(\bar{c}, w) < \beta\delta n/2$)[3], we have: $d_H(\tilde{c}, w) > \beta\delta n/2$ since $d \geq \beta\delta n$. Let w' be the noisy version of w. Consider the random variable $d_H(\tilde{c}, w')$. It is easy to see that $E\left(d_H(\tilde{c}, w')\right) \geq \delta n + \beta(1/2 - \delta)n$, so by Lemma (2.1), the probability $Pr[d_H(\tilde{c}, w') < \delta n + \beta\delta(1/2 - \delta)n - \kappa n]$ is exponentially small in n for all sufficiently small κ and all sufficiently large n. We conclude that if the cheating A tries to announce any codeword but the certain legal c which she has committed to (i.e., to change her mind) she will be accepted only with small probability. This proves the following Lemma.

Lemma 4.6 *Protocol 4.1 is binding for any $\kappa < \beta\delta(1/2 - \delta)/2$, i.e., the dishonest A fails to open any string b but the one she has committed to except with probability negligible in n.*

The completeness is argued taking into account that the honest A always sends a codeword to B. Let us consider the random variable $d_H(c, c')$. It is clear that $E\left(d_H(c, c')\right) = \delta n$, then by Lemma (2.1), $Pr\left[d_H(c, c') > (\delta + \kappa)n\right]$ is negligible in n for all sufficiently small κ and all sufficiently large n. Hence the honest B accepts with high probability when A is honest, yielding:

Lemma 4.7 *The completeness property fails only with probability negligible in n when $\kappa < \beta\delta(1/2 - \delta)/2$.*

4.4.4 Non-Asymptotic Analysis

In our non-asymptotic analysis, we shall use the BCH codes as discussed in Section 4.2.2.

The failure probabilities are defined as follows:

- P_c is the probability of completeness failure, i.e., the probability that honest A is rejected by honest B.
- P_A is the probability that binding fails, that is when dishonest A successfully changes her mind when opening the string b.
- P_B is dishonest B's advantage (compared to a random guess) in guessing b prior to the opening.

[3]Note that if there exists $\bar{c} \in C$, such that $d_H(\bar{c}, w) < \beta\delta n/2$ then we define A's commitment as $\bar{c}$. Otherwise, when A sends the word w which is at least $\beta\delta n/2$-far in Hamming sense from any codeword in C, A's commitment is defined as "?" as discussed in Remark 4.3

Lemma 4.8 *Completeness failure probability P_c can be expressed as follows:*

$$P_c = \sum_{i=d_t+1}^{n} \binom{n}{i} \delta^i (1-\delta)^{n-i}, \tag{4.12}$$

where d_t is the threshold such that B rejects if $d_H(c, c') > d_t$.

Proof. Follows by counting the probability that more than d_t errors occur during the transmission of c over the δ-BSC. □

Let r be the number of check bits of the chosen code C.

Lemma 4.9 *The dishonest B has a bias*

$$P_B = \tilde{P}_B - 1/2^l,$$

in guessing b (compared to a random guess) where $\tilde{P}_B$ is chosen according to

$$l + (1-\tilde{P}_B) \log \frac{1-\tilde{P}_B}{2^l - 1} + \tilde{P}_B \log \tilde{P}_B < I_b, \tag{4.13}$$

where

$$I_b \leq 2^{-(R-l-r)} / \ln 2, \tag{4.14}$$

$$R = -n \log \left(\delta^2 + (1-\delta)^2\right). \tag{4.15}$$

Proof. We bound the probability $\tilde{P}_B$ (which is the probability for the dishonest B to decode b correctly before the opening takes place) using Fano's inequality (2.3) as in (4.13) where I_b is his Shannon's information about b. The privacy amplification theorem gives us a bound (4.14) for I_b, while Equation (4.15) follows by the definition of Rényi entropy and its additivity. □

Lemma 4.10 *The dishonest A succeeds in violating the binding condition with the following probability:*

$$P_A = \max_{0 \leq s \leq d/2} (P[\textit{opening } c] + P[\textit{opening } \tilde{c}])/2, \textit{ where} \tag{4.16}$$

$$P[\textit{opening } c] = \sum_{i=0}^{s} \binom{s}{i} (1-\delta)^i \delta^{s-i} \sum_{j=0}^{d_t-i} \binom{n-s}{j} \delta^j (1-\delta)^{n-s-j}, \tag{4.17}$$

$$P[\textit{opening } \tilde{c}] = \sum_{i=0}^{d-s} \binom{d-s}{i} (1-\delta)^i \delta^{d-s-i} \sum_{j=0}^{d_t-i} \binom{n-d+s}{j} \delta^j (1-\delta)^{n-d+s-j}. \tag{4.18}$$

Proof. We recall first that the dishonest A may send to B not necessarily a codeword $c \in C$ but an arbitrary n-bit string w. B cannot immediately detect this fraud since w will be corrupted by noise. Suppose that A sends the string w

such that the nearest code word $c \in C$ occurs at the Hamming distance s from w, i.e. $d_H(c, w) = s$. Then, the following holds for some other code word $\tilde{c}$:

$$d_H(w, \tilde{c}) \geq d - s.$$

It is now easy to verify that the probabilities that A opens the codeword c or $\tilde{c}$ and they are accepted by B can be expressed as (4.17) and (4.18), respectively. In other words, these are the probabilities that sending a word which is s (or $d - s$) far from c in the Hamming sense does not add too much distortion to the noise introduced by the δ-BSC.

Now, (4.16) gives us the maximal probability for the cheating A to open successfully either c or $\tilde{c}$ ($c, \tilde{c} \in C$) while sending w (such that $d_H(w, c) = s$, $d_H(w, \tilde{c}) \geq d - s$) at the commitment stage. Clearly, this is the best that A can achieve. The claim now follows. □

The numerical calculations using Lemmas 4.8 – 4.10 give the following results.

Example 4.2 Fix $l = 1$, and the required failure probabilities $P_c = 10^{-4}$, $P_B = 10^{-4}$. Table 4.1 depicts the binding failure probability P_A which is computed for different code lengths n and the corresponding optimal error rate δ_{opt} (the rate for which P_A is minimal over all δ, $0 < \delta < 1/2$ and s, $0 \leq s \leq d/2$).[4]

Table 4.1: Binding failure probabilities

n	δ_{opt}	P_A
1023	0.213	0.738
2047	0.187	0.682
4095	0.179	0.268
8191	0.187	0.020
16383	0.169	$4.4 \cdot 10^{-5}$

The results of Example 4.2 show that the protocol provides relatively small failure probabilities in non-asymptotic case for relatively short n resulting in communication complexity of a few Kilobytes. For instance, the probability that A successfully changes her mind $P_A = 4.4 \cdot 10^{-5}$ for $n = 16383$, B decodes a codeword before the opening with probability $P_B = 10^{-4}$ (which is close to a random guessing) and completeness fails with probability $P_c = 10^{-4}$.

The other non-asymptotic result is worth mentioning. We analyse the *protocol rate* defined by $\alpha = l/n$ (the definition which is rigorously studied in [WNI03]), that characterise a "capacity" – its length l given the code length n, such that the protocol remains secure.

[4] In fact, $s = d/2$ always gave the minimal P_A.

Example 4.3 Fix $l = 20$ and let the requirements be $P_c = 10^{-4}$, $P_A = 5 \cdot 10^{-5}$ and $P_B = 10^{-6}$. Then the numerical calculations according to Lemmas 4.8 – 4.10 show that it suffices to choose a BCH code with $n = 16383$, $r = 8219$ and $d = 1175$.

This example shows that using hashing to the bit strings, one can obtain commitments to a (short) bit string "almost for the same price" as a commitment to a bit in terms of the security requirements.

4.4.5 Concluding Remarks

We conclude that the Bit Commitment protocol of [Cré97] can be made secure for the price of reasonably low communication complexity: of order of Kilobytes.

One can use the privacy amplification technique to commit to bit strings rather than one bit. In the bit-string case, there appears a trade-off between the protocol rate and the security requirements. One can use the relations presented in this section to optimise the parameters of the protocol given the security requirements.

Chapter 5

Universal Composability in Unconditional Model

5.1 Introduction

In this chapter, we introduce the *universally composable (UC)* framework of Canetti [Can01] modified for the unconditional model. Some preliminary ideas on secure protocol composition in this model appeared in [Can00][1] with application to secure function evaluation. Another related work of Ben-Or and Mayers [BM04] deals with UC in the quantum setting considering unconditional security as well.

In our presentation, we restrict ourselves to the case of two-party primitives based on noisy channels. This restriction helps to simplify the presentation and to focus on the issues which are essential for our setting. We believe although we do not prove it here formally, that the proposed modification is valid even without the restriction to a particular type of primitives or communication model. The material presented here is somewhat informal and not comprehensive since its main purpose is to introduce a basic notation and tools used for the UC security proofs in the next chapter. Nonetheless, we give a comparison to the original framework when it is relevant and argue that the original techniques and proofs are still valid under the changes we made.

We consider the protocols which take place in a model with two players A, B connected by an error-free channel and also by a noisy channel with some particular characteristic such as a UNC or a PassiveUNC. We assume a bounded delay in a message delivery for all channels such that failure to send a message can be detected. The noisy channels we study in this work can very conveniently be modelled as ideal functionalities and we assume that the players communicate using these functionalities but not an open network as in the original framework. Another related issue is a consequence of being in the two-party case: we do not think of our protocols as subroutines in a multi-player protocol, nor are we concerned with external observers, only with what a corrupted A or B might do or learn. We therefore assume that unless the adversary corrupts a player, he gets no information about the communication

[1] Although in this work, the label of *universal composability* had not yet been coined.

between A and B.

All our protocols are unconditionally secure for both players. The standard approach to security proofs has been (as in the previous chapters) to assume that either A or B is cheating, then prove some relevant security properties and if both parties are honest then they will accept the protocol (completeness property). We express it in the UC framework by assuming an infinitely powerful static adversary who from the onset has corrupted no one, or either A or B. The question whether our results extend to the case of adaptive adversaries is left out of the scope of our work.

Since the results we prove are information-theoretical in nature, we shall allow all the entities infinite computing power. At the same time, we stress that honest players can execute our protocols efficiently, i.e., with a number of subroutine invocations polynomial in the security parameter and the size of the input. This ensures that the composition theorem holds in this model if we demand a statistical simulation instead of a computationally indistinguishable one as in the original framework.

This presentation follows the lines of the excellent overview of Lindell [Lin03]. We refer the reader to [Can01a, Can03] for the details on the UC framework.

5.2 Overview of the Framework

We first review the syntax of protocols in our communication model. We present the real-life model of computation, the ideal process, and the general definition of UC realising an ideal functionality. Next, we present the hybrid model and composition theorem.

5.2.1 Protocol Syntax

Following [GMR85,Gol01], a protocol is represented as a system of probabilistic interactive Turing machines (ITM's), where each ITM represents the program to be run within a different party. Specifically, the input and output tapes model inputs and outputs that are received from and given to other programs running on the same machine, and the communication tapes model messages sent to and received from the network. Adversarial entities are also modelled as ITM's.

5.2.2 Basic Framework

Protocols that UC realise a given task (or, protocol problem) are defined in three steps, as follows. First, the process of executing a protocol in the presence of an adversary and in a given computational environment is formalised. This is called the *real-life model*. Next, an *ideal process* for carrying out the task at hand is formalised. In the ideal process the parties do not communicate with each other. Instead they have access to an "ideal functionality", which is essentially an incorruptible trusted party that is programmed to capture the desired functionality of the given task. A protocol is said to UC realise an ideal functionality if the process of running the protocol amounts to emulating the

ideal process for that ideal functionality. We overview the model for protocol execution (called the *real-life model*), the ideal process, and the notion of protocol emulation.

In the work [Can01], the players communicate over an asynchronous network which models current realistic communication networks (such as the Internet). But our case is different – we assume the players to be connected by both noisy and error-free channel which are modelled by ideal functionalities. Furthermore, we assume a bounded delay in message delivery for all channels such that failure to send a message can be detected. We note that an issue of authentication is not essential in the two-party case: upon receiving a message, a receiver knows who sent it. Parties may be broken into (i.e., become corrupted) throughout the computation, and once corrupted their behaviour is arbitrary (or, *malicious*). We restrict ourselves to the case of *static* adversary who has to decide on the onset of the protocol whether he will corrupt either one of the players or none. In Subsection 5.2.7, we discuss a modification to the general model for the case of *passive* (or *semi-honest*) adversary. Finally, we do not impose any restrictions on the computational power of all the involved entities.

5.2.3 Protocol Execution in the Real-Life Model

We sketch the process of executing a given protocol π (run by the players A and B) with some adversary Adv and an environment machine Z with input z. All entities have a security parameter $k \in \mathbf{N}$. The execution consists of a sequence of *activations*, where in each activation a single participant (either Z, Adv, or some $Q \in \{A, B\}$) is activated. The environment is activated first. In each activation, it may read the contents of the output tapes of all the uncorrupted parties[2] and the adversary, and may write information on the input tape of one of the parties or of the adversary. Once the activation of the environment is complete (i.e., once the environment enters a special waiting state), the entity whose input tape was written on is activated next.

The adversary may decide to corrupt a party on the onset of the protocol. Upon corrupting a party, the adversary gains access to all the tapes of that party and controls all the party's future actions. In addition, whenever a party is corrupted the environment is notified (say, via a message that is added to the output tape of the adversary). Once the adversary is activated, it may read its own tapes but not the outgoing communication tapes of the uncorrupted parties. This models our assumption that the adversary does not intercept the communication between the parties unless one of the latter is corrupted. Once the activation of the adversary is complete, the environment is activated next.

Once a party is activated due to an input given by the environment, it follows its code and possibly writes local outputs on its output tape. Once the activation of the party is complete the environment is activated. The protocol execution ends when the environment completes an activation without writing on the input tape of any entity. The output of the protocol execution is the

[2]The environment is not given read access to the corrupted parties output tapes because once a party is corrupted, it is no longer activated. Rather, the adversary sends messages in its name. Therefore, the output tapes of corrupted parties are not relevant.

output of the environment. We assume that this output consists of only a single bit.

In summary, the order of activations is as follows. The environment Z is always activated first. The environment then either activates the adversary Adv or some party Q by writing on an input tape. If the adversary is activated, it may return control to the environment, or it may activate some party Q by delivery a message to Q. After Q is activated, control is always returned to Z. We stress that at any point, only a single party is activated. Furthermore, Z and Adv can only activate one other entity (thus only a single input is written by Z per activation and likewise Adv can deliver only one message per activation).

Let $\text{REAL}_{\pi,Adv,Z}(k, z, \bar{r})$ denote the output of environment Z when interacting with adversary Adv and parties running protocol π on security parameter k, input z and random tapes $\bar{r} = r_Z, r_{Adv}, r_1, \ldots, r_n$ as described above (z and r_Z for Z, r_{Adv} for Adv; r_i for party P_i). Let $\text{REAL}_{\pi,Adv,Z}(k, z)$ denote the random variable describing $\text{REAL}_{\pi,Adv,Z}(k, z, \bar{r})$ when $\bar{r}$ is uniformly chosen.

5.2.4 Ideal Process

Security of protocols is defined via comparing the protocol execution in the real-life model to an *ideal process* for carrying out (a single instance of) the task at hand. A key ingredient in the ideal process is the *ideal functionality* that captures the desired functionality, or the specification, of the task. The ideal functionality is modelled as another ITM that interacts with the environment and the adversary via a process described below. More specifically, the ideal process involves an ideal functionality F, an *ideal process adversary* (or *simulator*) S, an environment Z with input z, and the *dummy parties* $\tilde{A}$, $\tilde{B}$.

As in the process of protocol execution in the real-life model, the environment is activated first. As there, in each activation it may read the contents of the output tapes of both (dummy) parties and the adversary, and may write information on the input tape of either one of the (dummy) parties or of the adversary. Once the activation of the environment is completed the entity whose input tape was written on is activated next.

The dummy parties are fixed and simple ITM's: Whenever a dummy party is activated with an input, it writes it on its outgoing communication tape for the ideal functionality F. Furthermore, whenever a dummy party is activated due to delivery of some message (from F), it copies this message to its output. At the conclusion of a dummy party's activation, the environment Z is activated. The communication by the dummy parties is with the ideal functionality F only. The messages sent between the dummy parties and F are secret and cannot be read by the adversary S. However, some information may be available to S and we demand that the functionality definition should specify it explicitly. For example, consider a commit message of the following format: (Commit, cID, b), where "commit" states that the party is committing to a new value, cID is the commitment identifier, b is the value being committed to. We call the issuer of the message *a committer* and the other party *a receiver*. In principle, the message must contain a so-called *session ID* as well, we leave it implicit so far and discuss this issue below in Subsection 5.2.8. Hence, in our example, the

functionality for Bit Commitment records cID, b and sends (Commit,cID) to the receiver and the adversary.

When the ideal functionality F is activated, it reads the contents of its incoming communication tape, and potentially sends messages to the parties and to the adversary by writing these messages on its outgoing communication tape. Once the activation of F is complete, the environment Z is activated next.

When the adversary S is activated, it may read only its own input tape. S cannot read the messages on the outgoing communication tape of F (unless the recipient of the message is S itself or a corrupted party). Then, S may write a message from itself on F's incoming communication tape.[3] On the onset of the protocol, the adversary S may corrupt a party. Upon corrupting a party, both Z and F learn the identity of the corrupted party (say, a special message is written on their respective incoming communication tapes).[4] The adversary controls the party's actions from the time that the corruption takes place. The adversary S may deliver a message to F on behalf of a corrupted party Q by copying it from Q's outgoing communication tape to Z's incoming communication tape.

If the adversary delivered a message to to the functionality F in an activation, then F is activated once the activation of S is complete. Otherwise the environment Z is activated next.

As in the real-life model, the protocol execution ends when the environment completes an activation without writing on the input tape of any entity. The output of the protocol execution is the (one bit) output of Z.

In summary, the order of activations in the ideal model is as follows. As in the real model, the environment Z is always activated first, and then activates either the adversary S or some dummy party P_i by writing an input. If the adversary S is activated, then it either activates a dummy party P_i or the ideal functionality F by delivering the entity a message, or it returns control to the environment. After the activation of a dummy party or the functionality, the environment is always activated next. Let $\text{IDEAL}_{F,S,Z}(k, z, \bar{r})$ denote the output of environment Z after interacting in the ideal process with adversary S and ideal functionality F, on security parameter k, input z, and random input $\bar{r} = r_Z, r_S, r_F$ as described above (z and r_Z for Z, r_S for S; r_F for F). Let $\text{IDEAL}_{F,S,Z}(k, z)$ denote the random variable describing $\text{IDEAL}_{F,S,Z}(k, z, \bar{r})$ when $\bar{r}$ is uniformly chosen.

[3] Many natural ideal functionalities indeed send messages to the adversary S (see, e.g., the BC functionality of Section 6.2.2). On the other hand, having the adversary S send messages to F is useful in order to reflect the (allowed) adversary's influence on the protocol or primitive that realises the functionality. For example, in Unfair Noisy Channel, the error rate may be chosen by the adversary, so we allow him this capability by having him send a value of the desired error rate to F. We stress that in our model, S is allowed to do that even if no player is corrupt. This models the deviations of channel's error rate which may happen in a realistic communication scenario.

[4] Allowing F to know which parties are corrupted gives it considerable power. This power provides greater freedom in formulating ideal functionalities for capturing the requirements of given tasks. On the other hand, it also inherently limits the scope of general realisability theorems.

5.2.5 UC Realising an Ideal Functionality

We say that a protocol π *UC realises* an ideal functionality F if for any real-life adversary Adv there exists an ideal process adversary S such that no environment Z, on any input, can tell with non-negligible probability whether it is interacting with Adv and parties running π in the real-life process, or with S and F in the ideal process. This means that, from the point of view of the environment, running protocol π is just as good as interacting with an ideal process for F. (In a way, Z serves as an interactive distinguisher between the two processes.) We have:

Definition 5.1 *Let $n \in \mathbf{N}$. Let F be an ideal functionality and let π be a two-party protocol. We say that π* UC realizes *F if for any adversary Adv there exists an ideal-process adversary S such that for any environment Z,*

$$\{IDEAL_{F,S,Z}(k,z)\}_{k\in\mathbf{N},z\in\{0,1\}^*} \simeq \{REAL_{F,S,Z}(k,z)\}_{k\in\mathbf{N},z\in\{0,1\}^*} \tag{5.1}$$

We remind that $\simeq$ here means statistical indistinguishability of the two ensembles.

5.2.6 On Non-Blocking Adversaries

Note that in the two-party setting, the definition provides no guarantee that a protocol will ever generate output or "return" to the calling protocol. Indeed, nothing prevents the adversary from corrupting a player and then refusing to continue the protocol. Instead, we concentrate on the security requirements *in the case that the protocol generates output.*

A corollary of the above fact is that a protocol that "hangs", never sends any messages and never generates output, UC realizes any ideal functionality. Thus, in order to obtain a meaningful feasibility result, we introduce the notion of a *non-blocking* simulator. This property means that if the real-life adversary does not corrupt any party then the simulator does not corrupt any party either. In this case, a party may not necessarily receive output. However, this only happens if either the functionality does not specify output for this party, of if the real-life adversary actively interferes in the execution by corrupting a party and refusing to continue interaction.

Note that demanding a simulator to be non-blocking is a way to state in the UC framework the traditional completeness property for a two-party protocol: if both players are honest, the protocol must complete successfully. In our protocols, we ensure that the simulators are non-blocking.

5.2.7 On Passive Adversaries

Definition 5.1 gives the adversary complete control over corrupted parties (such an adversary is called *active*). Specifically, the model states that from the time of corruption the corrupted party is no longer activated, and instead the adversary sends messages in the name of that party. In contrast, when a *passive* adversary corrupts a party, the party continues to follow the prescribed protocol.

Nevertheless, the adversary is given read access to the internal state of the party at all times. Everything else remains the same as in the above-described active model. We say that protocol π UC realizes functionality F *for passive adversaries*, if for any passive real-life adversary Adv there exists an ideal-process passive adversary S such that Equation 5.1 holds for any environment Z.

5.2.8 Composition Theorem

In order to state the composition theorem, and in particular in order to formalise the notion of a real-life protocol with access to multiple copies of an ideal functionality, the *hybrid model of computation* with access to an ideal functionality F (or, in short, the F-hybrid model) is formulated. This model is identical to the real-life model, with the following additions. On top of sending messages to each other, the parties may send messages to and receive messages from an unbounded number of copies of F. Each copy of F is identified via a unique *session identifier* (sID); all messages addressed to this copy and all message sent by this copy carry the corresponding sID. The session ID's play a central role in the hybrid model and the composition operation, in that they enable the parties to distinguish different instances of a protocol. Indeed, differentiating protocol instances via session ID's is a natural and common mechanism in protocol design.

However, it turns out that among the functionalities which we shall introduce in the next chapter, an explicit treatment of the session identifiers is essential only in one of them (the functionality RandomChoice defined in Subsection 6.2.3). Therefore, we shall explicitly handle the session ID's only in the definition of the RandomChoice leaving this issue implicit for the rest of functionalities. We assume that the session ID's are handled there in the standard way.

The communication between the parties and each one of the copies of F mimics the ideal process. That is, when the message is delivered from a party to a copy of F with a particular sID, that copy of F is the next entity to be activated. (If no such copy of F exists then a new copy of F is created and activated to receive the message.)

The hybrid model does not specify how the sID's are generated, nor does it specify how parties "agree" on the sID of a certain protocol copy that is to be run by them. These tasks are left to the protocol in the hybrid model. This convention simplifies formulating ideal functionalities, and designing protocols that UC realise them, by freeing the functionality from the need to choose the sID's and guarantee their uniqueness. In addition, it seems to reflect common practice of protocol design in existing networks.

Let $\text{HYBRID}^{F}_{\pi,Adv,Z}(k,z)$ denote the random variable describing the output of environment machine Z on input z, after interacting in the F-hybrid model with protocol π and adversary Adv, analogously to the definition of $\text{REAL}_{\pi,Adv,Z}(k,z)$. (We stress that here π is a hybrid of a real-life protocol with ideal evaluation calls to F.)

5.2.9 Replacing a Call to F with a Protocol Invocation

Let π be a protocol in the F-hybrid model, and let ρ be a protocol that UC realizes F. The composed protocol π^ρ is constructed by modifying the code of each ITM in π so that the first message sent to each copy of F is replaced with an invocation of a new copy of ρ with fresh random input, with the same sID, and with the contents of that message as input. Each subsequent message to that copy of F is replaced with an activation of the corresponding copy of ρ, with the contents of that message given to ρ as new input. Each output value generated by a copy of ρ is treated as a message received from the corresponding copy of F. (See [Can01] for more details on the operation of composed protocols, where a party, i.e., an ITM, runs multiple protocol-instances concurrently.) If protocol ρ is a protocol in the real-life model then so is π^ρ. If ρ is a protocol in some G-hybrid model (i.e., ρ uses ideal evaluation calls to some functionality G) then so is π^ρ.

5.2.10 Theorem Statement

In its general form, the composition theorem basically says that if ρ UC realizes F in the G-hybrid model for some functionality G, then an execution of the composed protocol π^ρ, running in the G-hybrid model, "emulates" an execution of protocol π in the F-hybrid model. That is, for any adversary Adv in the G-hybrid model there exists an adversary S in the F-hybrid model such that no environment machine Z can tell with non-negligible probability whether it is interacting with Adv and π^ρ in the G-hybrid model or it is interacting with S and π in the F-hybrid model.

A corollary of the general theorem states that if π UC realises some functionality I in the F-hybrid model, and ρ UC realises F in the G-hybrid model, then π^ρ UC realises I in the G-hybrid model. (Here one has to define what it means to UC realise functionality I in the F-hybrid model. This is done in the natural way.) That is:

Theorem 5.1 ([Can01]) *Let F, G, I be ideal functionalities. Let π be an n-party protocol in the F-hybrid model, and let ρ be an n-party protocol that UC realises F in the G-hybrid model. Then for any adversary Adv in the G-hybrid model there exists an adversary S in the F-hybrid model such that for any environment machine Z we have:*

$$\left\{HYBRID^G_{\pi^\rho,Adv,Z}(k,z)\right\}_{k\in\mathbf{N},z\in\{0,1\}^*} \simeq \left\{HYBRID^F_{\pi,S,Z}(k,z)\right\}_{k\in\mathbf{N},z\in\{0,1\}^*}$$

In particular, if π UC realises functionality I in the F-hybrid model then π^ρ UC realises I in the G-hybrid model.

Consider the case that G is the empty functionality, and so the G-hybrid model is actually the *real model*. Then, Theorem 5.1 states that ρ remains secure when run concurrently with *any* protocol π. In other words, ρ remains secure under concurrent general composition (with arbitrary sets of parties). We note that although π technically seems to be a "calling protocol", it can also represent arbitrary network activity. Thus, we obtain that ρ remains secure when run concurrently in an arbitrary network.

5.3 On the Proof of Composition Theorem

The main concern in our modification of the UC framework is to make sure that the proof of the UC composition theorem still works in this case. We shall give an intuition why this is true. Recall the proof of the composition theorem (Section 5.4 of [Can01a]). Let F be an ideal functionality, π be a protocol in the F-hybrid model and ρ be a protocol that securely realises F, let π^ρ be the composed protocol. The aim is to construct an adversary H in the F-hybrid model such that no Z will be able to tell whether it is interacting with π^ρ and adversary Adv in the real-life model or with π and H in the F-hybrid model. That is, for any Z, H should satisfy

$$\mathrm{REAL}_{\pi^\rho,Adv,Z} \simeq \mathrm{HYBRID}^F_{\pi,H,Z}. \tag{5.2}$$

The security of ρ guarantees that there exists a simulator S, such that for any environment Z_ρ:

$$\mathrm{REAL}_{\rho,Adv,Z_\rho} \simeq \mathrm{IDEAL}_{F,S,Z}. \tag{5.3}$$

The construction of H is given in [Can01a] and its validity is demonstrated, based on validity of S, via a so-called *hybrid argument*. Let $m = m(k)$, where k is the security parameter, be an upper bound on the number of copies of ρ that are invoked in this interaction. Informally, we let the $F^{(l)}$-hybrid model denote the model where the interaction with the first l copies of F works as in the F-hybrid model, whereas the rest of the copies of F are replaced with an invocation of ρ. In particular, an interaction of Z with H in the $F^{(m)}$-hybrid model is essentially identical to an interaction of Z with H in the F-hybrid model. Similarly, an interaction of Z with H in the $F^{(0)}$-hybrid model is essentially identical to an interaction of Z with Adv in the real-life model.

Now, assume that there exists an environment Z that distinguishes with probability ϵ between an interaction with H in the F-hybrid model and an interaction with Adv in the real-life model. Then there is an $0 \leq l \leq m$ such that Z distinguishes between an interaction with H in the $F^{(l)}$-hybrid model and an interaction with H in the $F^{(l-1)}$-hybrid model. We then construct an environment Z_ρ, that interacts with parties running a single copy of ρ, and can distinguish with probability ϵ/m between an interaction with Adv and parties running ρ in the real-life model, and an interaction with S in the ideal process for F. Clearly, this would violate Equation (5.3).

The important observation is that $m(k)$ is polynomial in k since the ITM's are PPT in the original framework. Therefore, no matter how many copies of ρ are run, their "total contribution" will not be enough for the environment Z_ρ to distinguish successfully, so the proof by contradiction is possible. In our case, the involved entities are unbounded in their computing power however, we stress that our protocols are efficient, i.e., the honest players can complete them in number of steps polynomial in the security parameter k. Hence, they are not going to run more than polynomially many copies of ρ and exactly the same argument as in [Can01] (but for statistical indistinguishability) will be valid in our modification as well.

Chapter 6

Weakened Assumption: Unfair Noisy Channel

6.1 Introduction

In this chapter, we consider an important issue in unconditional cryptographic primitives – making the very assumption, i.e., the noise model more practical. In the previous chapters, we assumed that the players knew the distribution of the noise *precisely* and could not affect it, while such the influence is quite possible in the real channels, for instance, in an open-air communication such as radio-links.

Here, we elaborate on the methods and techniques introduced by Damgård, Kilian and Salvail in [DKS99] for implementing OT based on Unfair Noisy Channels (UNC). In this model, the randomness introduced by UNC may be partially controlled by the corrupted player.

We present a generic "compiler" which transforms any protocol implementing OT from a passive version of UNC and secure against passive cheating into a protocol that uses UNC for communication and builds an OT secure against active cheating. We argue the new result in terms of the UC framework modified for the unconditional model as described in the previous chapter. Our construction fixes a flaw in the proof of [DKS99] which was incomplete otherwise. We exploit this result and a new technique for transforming between the weaker versions of Oblivious Transfer, in order to prove a stronger positive OT result then the one claimed in [DKS99].

The material of this chapter is based on [DFMS04].

6.2 Preliminaries

In this chapter, we refer to a value which decreases faster than any polynomial fraction in the security parameter k as to *negligible* in k.

6.2.1 Communication Model

We assume that the players are connected by the unfair noisy channel (see the definition below) and in addition, the error-free channel. For the sake of

simplicity, we also assume that the adversary does not intercept the error-free channel, i.e., he learns nothing about the players' conversation unless he corrupts either of them.

The assumption that a corrupted player has partial control over the randomness introduced in the noisy channel implies that he can gain some side information which in general may not be accessible to an honest party. As a matter of fact, we always assume that it is only an adversary who can use this side information. We shall call this extra information the *unfair advantage*.

We shall define the error-free and the unfair noisy channels as ideal functionalities which the players will use for conversations. In order to model our communication setting, we assume that whenever a player transmits any data through the channel in the real world, he uses the corresponding functionality to do so.

The functionality EFC for the error-free channel is defined in the following way:

Functionality EFC

Send b**:** The issuer of this command is called a sender, the other party is a receiver. On receipt of this command where $b \in \{0,1\}$, the functionality sends b to the receiver.

A (γ,δ)-Unfair Noisy Channel ((γ,δ)-UNC) is specified by the following functionality:

Functionality (γ,δ)-UNC

Send b**:** The issuer of this command is called a sender, the other party is a receiver. On receipt of this command where $b \in \{0,1\}$, the functionality records b and outputs a string "which error probability?" to the adversary. It ignores all further commands until the adversary sends an Error probability command.

Error probability ϵ**:** Receiving this command from the adversary, the functionality checks whether $\gamma \leq \epsilon \leq \delta$. If not, the command is ignored. Otherwise, it chooses a random bit b', such that $P[b'=1]=\epsilon$, and sends $\hat{b} = b \oplus b'$ to the receiver.

What we want to model here is intuitively that a corrupted player may influence the error rate or even block the channel. But if both players are honest, transmissions will always go through, however, the error rate will fluctuate in some arbitrary way in the given interval. We therefore assume throughout about the adversary that if both players are honest, then the adversary will always give a legal error probability back when receiving a request from the UNC.

Remark 6.1 As mentioned, the adversary is allowed to set the error probability to any value in $[\gamma..\delta]$ for every transmission. However, if the adversary corrupts a player, any attack he can do following, say, algorithm Alg can be simulated perfectly by an adversary that sets the error rate to γ always, but

adds artificial noise to any bit sent (or received) in case Alg wanted a larger error rate. We may therefore assume that an active adversary who corrupts A or B always sets the error rate of the UNC to γ.

A (γ, δ)-Passive Unfair Noisy Channel ((γ, δ)-PassiveUNC) models the scenario where the adversary cannot physically influence the channel, so if the players are honest then the channel is a BSC with error rate δ. However, a corrupted player (which cheats passively) gets some unfair advantage which allows him to reduce the error rate to γ *from his point of view*. For instance, in a scenario with a radio-link between the players, it models the cheater who obtains a more sensitive receiving equipment.

Functionality (γ, δ)-PassiveUNC

Send b**:** The issuer of this command is called a sender, the other party is a receiver. On receipt of this command where $b \in \{0, 1\}$, the functionality chooses random bits b', b'', such that $P[b' = 1] = \gamma$ and $P[b'' = 1] = \nu$, where $\nu \boxplus \gamma = \delta$.[1] The functionality sends $\hat{b} = b \oplus b' \oplus b''$ to the receiver. If the adversary has corrupted a player, it sends to the adversary a bit z, where $z = b \oplus b''$ if the sender is corrupted, and $z = b \oplus b'$ if the receiver is corrupted.[2]

Remark 6.2 In the future discussions, if the noisy channel is a UNC, then we assume the adversary to be active, i.e., he can decide the corrupted player's behaviour. If the channel is a PassiveUNC, the adversary is passive.

6.2.2 Functionalities for Basic Primitives

We start by describing the ideal functionalities for Oblivious Transfer and Bit Commitment.

Functionality OT

Send b_0, b_1**:** The issuer of the Send command is called the sender, the other party is the receiver. On receipt of this command where $b_0, b_1 \in \{0, 1\}$, the functionality records b_0, b_1 and outputs "which bit?" to the receiver. It ignores all further commands until the receiver sends a Choice command.

Choice c**:** Receiving this command from the receiver, the functionality sends b_c to the receiver if $c \in \{0, 1\}$ and otherwise ignores the command.

Bit commitment can be modelled by an ideal functionality where one commits by giving the bit to the trusted party, who will then later open it on request from the committer.

[1] This ensures that $P[b' \oplus b'' = 1] = \delta$ according to (2.4).

[2] Intuitively, given z, the noise rate goes down to γ from the adversary's point of view.

Functionality BC

Commit cID, b**:** The issuer of this command is called the committer, the other player is the receiver. Receiving this command, where cID is a bit-string[3] and b is a bit, do as follows: if no message containing cID has been received yet, record the value of cID, b and send $(Commit, cID)$ to the receiver and the adversary Adv.

Open cID**:** If cID, b has been received earlier from the player issuing this command, send b to the other player and Adv.

6.2.3 More Functionalities

We shall use bit commitments and zero-knowledge proofs in our protocols. Those are modelled by an ideal functionality which is analogous to the functionality BC described in the previous chapter, but furthermore, the trusted party will confirm that the committed bits satisfy a given formula, if this is indeed the case.

Functionality Commit-and-prove (CaP)

Commit cID, b**:** The issuer of this command is called the committer, the other player is the receiver. Receiving this command, where cID is a bit-string and b is a bit, do as follows: if no message containing cID has been received yet, record the value of cID, b and send as output $(Commit, cID)$ to the receiver and the adversary.

Open cID**:** If cID, b has been received earlier from the player issuing this command, send b to the receiver and the adversary.

Prove L, Φ**:** Receiving this command, where L is a list of bit strings and Φ is a Boolean formula, check if L contains only strings that has been used as identifiers for bits committed to by the issuer of the Prove command. If so, find the corresponding bits and check whether they satisfy Φ. If so, send (OK, L, Φ) otherwise, send $(Fail, L, \Phi)$ to the receiver and the adversary.

As bit commitment scheme in our protocols, we shall use the UNC-based construction from [DKS99], which works assuming $\delta < 2\gamma(1-\gamma)$ which we shall assume throughout.[4] This scheme is unconditionally secure for both players. Furthermore, given any commitment scheme, one can always construct a new one, where one can prove in zero-knowledge that committed bits satisfy a given Boolean formula (see [BGGHKMR88]). It follows that in any protocol where we assume access to the UNC functionality, we may assume also CaP without loss of generality.

One more functionality which will come in handy is the ability to choose random bits and numbers with a prescribed distribution:

[3] We shall later refer to it as an *identifier* for the bit committed to.

[4] Since otherwise, the (γ, δ)-UNC appears to be trivial that is argued in the next subsection.

Functionality RandomChoice

Flip ν**:** Here, sID is a session ID and ν must be a probability. Once the functionality has received this command from both players containing identical values of sID, ν, it chooses a bit b at random such that $P[b = 1] = \nu$ and sends b to both players.

Uniform sID, j**:** Here sID is a session ID and j must be a natural number. Once the functionality has received this command from both players containing identical values of sID, j, it chooses i uniformly from $[0..j-1]$ and sends i to both players.

Using standard techniques, one can implement this functionality based on the CaP, with a statistically good simulation. It should be noted that in our two-player scenario, functionalities such as RandomChoice can only be realised if the adversary is allowed to abort after seeing the output. But this is consistent with the UC framework, where adversary and simulator are indeed allowed to abort at any time.

In [DKS99], a general model for two-party primitives is proposed where a corrupted player can gain more information than an honest one, this is called *Weak Generic Transfer* (*WGT*). We shall not go into details about this model, we only mention its important advantage: it accommodates the possibility for *either* player to obtain extra information. In contrast, Brassard and Crépeau [BC97], and Cachin [Cac98] consider the models where this weakness is one-sided, i.e., only the receiver can gain some side information.

We introduce next a particular case of WGT, a faulty version of OT: *Weak Oblivious Transfer* (*WOT*). This primitive comes in handy when attempting to build OT from UNC.[5] A (p, q, ϵ)-Weak Oblivious Transfer is defined in [DKS99] as OT with the following faults: with probability at most p a cheating sender A learns the selection bit c; with probability at most q a cheating receiver B learns both bits b_0, b_1; an honest B gets the incorrect bit, i.e., $1 - b_c$ with probability at most ϵ.

Some important reductions among these primitives and impossibility results were studied in [DKS99]. We shall briefly sketch some of them below in order to make this chapter self-containing. We stress that the aforementioned reductions and models are of general interest, independently of particular implementation but nevertheless in our work, we shall exploit them only as a tool allowing us to achieve OT using UNC's.

We first introduce the functionality for Weak Oblivious Transfer.

Functionality (p, q, ϵ)-WOT

Send b_0, b_1**:** The functionality's action on this command is the same as in OT.

Choice c**:** If $c \notin \{0, 1\}$ then the functionality ignores the command. Otherwise, it chooses $\tilde{b}_c \in \{0, 1\}$ such that $P[\tilde{b}_c \neq b_c] = \epsilon$ and sends it to the receiver. Additionally, if the sender is corrupted, then with probability p it sends

[5]Note that the Unfair Noisy Channels (being the "weak" versions of Binary Symmetric Channel) are also a particular case of WGT.

c to the sender, and if the receiver is corrupted, then with probability q it sends b_{1-c} to the receiver.

6.2.4 Impossibility Results

The notion of *triviality* for unfair noisy channels was introduced in [DKS99]. We shall call an unfair noisy channel *trivial* (or *simulatable*), if no unconditionally secure (for both parties) primitive can be build from it. Consequently, a natural way of arguing triviality of some primitive is using the symmetry condition discussed in Subsection 2.9.1. Clearly, the primitive is trivial, if it can be reduced to another primitive where only a noiseless communication and a local randomness are available to the parties. We shall assume from now on that γ and δ are both less than $1/2$ since otherwise no information transmission is possible for honest players.

Lemma 6.1 [DKS99] *(γ, δ)-PassiveUNC is trivial when $\delta \geq 2\gamma(1-\gamma)$.*

Proof. First, we are going to "implement" (γ, δ)-PassiveUNC with $\delta = 2\gamma(1-\gamma)$ using only error-free channel and local coin flips. Let the sender A has an input bit b.

Protocol 6.1 SimPassiveUNC$[\gamma](b)$

1. A and B pick the bits b_A and b_B respectively, such that $P[b_A = 1] = P[b_B = 1] = \gamma$.
2. A announces $b' = b \oplus b_A$ to B. B computes $b^* = b' \oplus b_B$, denoting b^* as the received bit, while no output is defined for A.

It is easy to show that the reduction SimPassiveUNC$[\gamma](b)$ indeed implements the (γ, δ)-PassiveUNC. Since A can cheat only passively, she may save b' and this is a side information which allows her to reduce the error rate down to γ because $Pr[b' \neq b^*] = \gamma$. However, for the honest A, who ignores this side information the error rate remains equal to $P[b \neq b^*] = 2\gamma(1-\gamma) = \gamma \boxplus \gamma = \delta$. The analogous reasoning is true for B.

Moreover, the same argument as above carries over for the case $\delta' > 2\gamma(1-\gamma)$ because one can instruct the parties to flip their bits with probabilities γ' such that $\delta' = \gamma' \boxplus \gamma'$. □

It is not known whether it is possible to construct a similar "implementation" of (γ, δ)-UNC, since active cheating is possible there and hence it seems to be difficult to make sure that the players flip their bits with proper probabilities. The way around this problem was suggested in [DKS99], the basic idea was to observe that (γ, δ)-UNC is a strictly weaker assumption than (γ, δ)-PassiveUNC and therefore if it is impossible to build a secure OT from (γ, δ)-PassiveUNC for $\delta \geq 2\gamma(1-\gamma)$, it will be impossible based on (γ, δ)-UNC either.

Lemma 6.2 [DKS99] *(γ, δ)-UNC is trivial when $\delta \geq 2\gamma(1-\gamma)$.*

Proof. Suppose that there exists a reduction from a primitive P to a (γ, δ)-UNC where $\delta = 2\gamma(1-\gamma)$ which is secure against active attacks. Let us compare the following two cases. In Case 1, the reduction is attacked by an adversary using the following active cheating strategy for a player $Q \in \{A, B\}$: Q sets the error rate to γ always, and then does the following: whenever Q is supposed to send a bit, Q first flips it with probability γ and then actually sends it. Similarly, whenever Q receives a bit from the channel, Q flips it with probability γ and acts as if that was the bit actually received. In the other cases, Q follows the algorithm specified by the reduction. Case 2: we execute the algorithm of P substituting the (γ, δ)-UNC by a (γ, δ)-PassiveUNC, and the adversary executes a passive attack.

It is easy to see that there is no difference between the cases from the honest player's point of view. Observe that in Case 1, the adversary following the strategy for Q knows as much about every bit sent and received by his opponent as a passive adversary knows in Case 2. Since by the assumption, the reduction is secure in Case 1, it must be secure in Case 2, a contradiction.

The same argument is valid for $\delta > 2\gamma(1-\gamma)$ taking into account Lemma 6.1. □

Throughout this chapter, we are going to assume a non-trivial (γ, δ)-UNC (i.e., with $\delta < 2\gamma(1-\gamma)$) connecting the two players.

Lemma 6.3 [DKS99] *(p, q, ϵ)-WOT is trivial when $p + q + 2\epsilon \geq 1$.*

Proof. We implement (p, q, ϵ)-WOT where $2\epsilon = 1 - p - q$ using an error-free channel. Let the sender A's input be b_0, b_1 and the receiver B's input be c.

Protocol 6.2 SimWOT$[p, q, \epsilon]((b_0, b_1), c)$

1. With probability q, A announces b_0, b_1, B computes b_c and the protocol terminates; otherwise A announces "pass".

2. If A passes, then with probability $p/(1-q)$, B sends c to A who replies with b_c; otherwise, B chooses b_c at random.

Let us argue that this protocol indeed implements the (p, q, ϵ)-WOT. We first assume $p + q + 2\epsilon = 1$. Clearly, B learns both b_0 and b_1 with probability q. A learns c with probability $p/(1-q)$ in Step 2, if she passed in Step 1 (this happens independently with probability $1-q$) that results in probability p. Finally, B receives an incorrect b_c, if he guesses incorrectly in Step 2, the probability that he has to guess at all is $(1 - p(1-q))(1-q)$, so his error probability is $\epsilon = (1-p-q)/2$.

The argument above is for $p + q + 2\epsilon = 1$. If $p + q + 2\epsilon > 1$, choose $\epsilon' = (1-p-q)/2 < \epsilon$; the impossibility result is true for (p, q, ϵ')-WOT. Note that a (p, q, ϵ')-WOT primitive also meets the requirements of a (p, q, ϵ)-WOT primitive since $\epsilon' < \epsilon$. Therefore, if OT is reducible to (p, q, ϵ)-WOT, it must also be reducible to (p, q, ϵ')-WOT, a contradiction. □

6.3 Possibility Results in the Passive Case

We first assume the players to be passive and build OT from (γ, δ)-PassiveUNC. The active cheating is dealt with in the next Section.

A straight forward idea would be to exploit a construction similar to the one used in Chapter 3. However, it is easy to see that Protocol 3.2 fails to provide security for B when the players use (γ, δ)-PassiveUNC for communication. A's side information allows her to gain a non-negligible bias for some non-negligible fraction of the sent bits when she tries to tell the good bits from the bad ones. Hence, B's privacy failure probability is always non-negligible in this case. A new reduction (Protocol 6.3) was therefore designed for (γ, δ)-PassiveUNC [DKS99]. Its output was not an original OT but its "imperfect" version where the privacy of the parties could still be violated with some non-negligible probabilities. The output of the reduction[6] was modelled as a (p, q, ϵ)-Weak Oblivious Transfer.

Protocol 6.3 WOTfromPassiveUNC((b_0, b_1),c)

1. A picks $x, y \in_R \{0, 1\}$,
2. A sends (xx, yy) through (γ, δ)-PassiveUNC and B receives $(\tilde{x}\tilde{x}', \tilde{y}\tilde{y}')$,
3. If B receives $(\tilde{x} \oplus \tilde{x}', \tilde{y} \oplus \tilde{y}') \notin \{(0, 1), (1, 0)\}$ then they go to step 1.
4. B announces w such that
   - $w = 0$ if $((\tilde{x} \oplus \tilde{x}' = 0) \wedge (c = 0)) \vee ((\tilde{y} \oplus \tilde{y}' = 0) \wedge (c = 1))$,
   - $w = 1$ if $((\tilde{x} \oplus \tilde{x}' = 0) \wedge (c = 1)) \vee ((\tilde{y} \oplus \tilde{y}' = 0) \wedge (c = 0))$,
5. A announces
   - $(a, b) = (x \oplus b_0, y \oplus b_1)$ if $w = 0$,
   - $(a, b) = (y \oplus b_0, x \oplus b_1)$ if $w = 1$,
6. B computes
   - $b_0 = a \oplus \tilde{x}$ if $c = 0$ and $w = 0$,
   - $b_0 = a \oplus \tilde{y}$ if $c = 0$ and $w = 1$,
   - $b_1 = b \oplus \tilde{y}$ if $c = 1$ and $w = 0$,
   - $b_1 = b \oplus \tilde{x}$ if $c = 1$ and $w = 1$.

The intuition behind the reduction is the following: In Steps 1-3, A is supposed to send two pairs of bits to B, until he receives an erasure and a bit, so A is sure that one of the bits x, y is completely lost for B. If we had a δ-BSC connecting the parties, we would obtain a (0,0,ϵ)-WOT, because A

[6] We shall refer to some protocols in this chapter as to "reductions" partly due to historical reasons and partly because these are indeed protocols which produce another protocols (or primitives) as their outputs. As a matter of fact, one can consider, for instance, Protocol 3.3 as the reduction of OT to Passive OT.

would have no idea which of the pairs was erased. On the other hand, B "tells" A which bit he wants to receive in Step 4 and A releases the bits in Step 5. If the players follow the protocol, B only receives an incorrect bit when the errors occur in both bits of the same pair in Step 2. This event contributes to the error probability ϵ.

But in fact, the channel connecting the players is PassiveUNC where the cheaters can actually obtain some side information about the data sent and received though they cannot affect the error rate of the channel directly. One way to model such the additional information is to assume that the corrupted player makes an auxiliary measurement "in the middle of the channel". For example, the cheating receiver receives a bit twice: "in the middle" – as over γ-BSC and then (the same bit) which passed "the rest of the cascade" (μ-BSC) – as over δ-BSC, where $\delta = \mu \boxplus \gamma$.

The failure probabilities p and q characterise the side information obtained by a corrupted player. We derive the dependencies $p(\gamma, \delta)$, $q(\gamma, \delta)$ and $\epsilon(\delta)$ later in Section 6.5 along with more advanced characterisation of side information and extended possibility results compared to the ones introduced in [DKS99].

6.3.1 Some Reductions

The following three reductions will be used for decreasing the failure probabilities of (p, q, ϵ)-WOT thereby obtaining original OT. The first is to reduce the sender's side information (decreasing p), the second is to cope with receiver's side information (decreasing q), while the third one is to reduce the error rate ϵ. It turns out that each reduction improves the intended parameter but increases the other two.

The reductions are assumed to be given as black-box protocol $\mathcal{W}$ implementing (p, q, ϵ)-WOT and take a security parameter k as an input. The reduction S-Red is taken from [CK88], R-Red is attributed to folklore in [DKS99] and E-Red is (kind of) a repetition code properly applied to b_c. In each reduction, we let b_0, b_1 be the input of A, and c be the input of B.

Protocol 6.4 S-Red(k_S)

1. $\mathcal{W}$ is executed k_S times, with inputs (b_{0i}, b_{1i}), $i = 1, \ldots, k_S$ for A and c_i, $i = 1, \ldots, k_S$ for B, where the b_{0i} are uniformly chosen, such that $b_0 = \oplus_{i=1}^{k_S} b_{0i}$, $b_{1i} = b_{0i} \oplus b_0 \oplus b_1$ and the c_i's are uniformly chosen, such that $c = \oplus_{i=1}^{k_S} c_i$.

2. B computes his output bit as follows: $b_c = \oplus_{i=1}^{k_S} b_{c_i}$.

Protocol 6.5 R-Red(k_R)

1. $\mathcal{W}$ is executed k_R times, with inputs (b_{0i}, b_{1i}), $i = 1, \ldots, k_R$ for A and c_i, $i = 1, \ldots, k_R$ for B, where $c_i = c$, $b_0 = \oplus_{i=1}^{k_R} b_{0i}$ and $b_1 = \oplus_{i=1}^{k_R} b_{1i}$.

2. B computes his output bit as follows: $b_c = \oplus_{i=1}^{k_R} b_{c_i}$.

Protocol 6.6 E-Red(k_E)

1. A picks $q_0, q_1 \in_R \{0,1\}$ and B picks $s \in_R \{0,1\}$,
2. A sends (q_0, q_1) k_E times through $\mathcal{W}$ to B and B selects the bit q_s k_E times,
3. If B did not receive the same bits q'_s all the k_E times then A and B go to Step 1.
4. B announces $y = 0$ if $s = c$ and $y = 1$ otherwise.
5. A announces r_0 and r_1 such that $b_y = r_0 \oplus q_0$, $b_{1-y} = r_1 \oplus q_1$, allowing B to compute $b_c = r_s \oplus q'_s$.

Lemma 6.4 [DKS99] *When given k and a (p, q, ϵ)-WOT $\mathcal{W}$ as input,*

- *S-Red(k) implements a $\big(p^k, 1-(1-q)^k, ((2\epsilon-1)^k+1)/2\big)$-WOT,*
- *R-Red(k) implements a $\big(1-(1-p)^k, q^k, ((2\epsilon-1)^k+1)/2\big)$-WOT,*
- *E-Red(k) implements a $\left(1-(1-p)^k, 1-(1-q)^k, \frac{\epsilon^k}{\epsilon^k+(1-\epsilon)^k}\right)$-WOT.*

All the three reductions produce a WOT secure against active cheating if the given WOT has this property.

Proof. It follows by inspection that the reductions allow the players to compute the correct output.

Let us consider the parameters of the obtained WOT's. In S-Red, the dishonest A learns c if and only if she learns all c_i's that happens with probability p^k. On the other hand, the dishonest B can learn both b_0 and b_1 if he gets just one pair (b_{0i}, b_{1i}) and this happens with probability $1-(1-q)^k$. When one XORs k bits each having error probability ϵ, the error rate of the resulting bit is equal to $((2\epsilon-1)^k+1)/2$ as argued in [Cré90], Proposition 4.6.

The case of R-Red is similar but with the chances of the sender and receiver reversed. As for E-Red, note that a WOT with the same inputs is used k times. Clearly, either dishonest A or dishonest B needs only one chance out of k to learn the private input of the other player, therefore both parameters p and q increase as in the above reductions. At the same time, the error rate improves to $\epsilon^k/(\epsilon^k+(1-\epsilon)^k)$ since an error must occur in all k bits q_s given that all the received bits q'_s are the same.

The argument for the last claim is the following: in S-Red, security of $\mathcal{W}$ means that none of the players can gain anything from inputting "?" to $\mathcal{W}$. And if indeed no "?" is input to any $\mathcal{W}$ instance, then B always behaves consistently with some input c, namely the value $c = \oplus_{i=1}^{k} c_i$. A can behave inconsistently by choosing random values of her input bits but this will not give her more information on c. The cases of R-Red and E-Red are similar. □

The next step is to apply the reductions S-Red, R-Red and E-Red in order to reduce OT to (p, q, ϵ)-WOT. The following lemma is proved in [DKS99].

Lemma 6.5 *There exists a sequence of reductions S-Red, R-Red and E-Red that implements OT given any (p, q, ϵ)-WOT that satisfies $p + q + 2\epsilon \leq 0.45$.*

The proof proceeds along the following lines: the reductions S-Red, R-Red and E-Red are combined and then the general expression for the parameters of the resulting WOT is obtained. Then, it is possible to to find certain parameters k_S, k_R, k_E and security parameter k which is the overall number of WOT's invocations, such that the sum $p + q + 2\epsilon$ for each produced WOT decreases monotonously and converges to 0 exponentially fast in k.

Recall that Protocol 6.3 produces a $(p(\gamma, \delta), q(\gamma, \delta), \epsilon(\delta))$-WOT $\mathcal{W}$ therefore, by Lemma 6.5, we have OT secure against passive cheating for some range of (γ, δ).

6.4 Protection Against Active Cheating

In this section, we consider a generic "compiler" which transforms any protocol implementing OT from (γ, δ)-PassiveUNC and secure against passive cheating into a protocol that uses (γ, δ)-UNC for communications and builds an OT secure against active cheating.

The existence of such the compiler implies a rather surprising fact that a passive adversary is essentially as powerful as an active one with respect to OT based on Unfair Noisy Channels.

6.4.1 Committed PassiveUNC

We first define informally the notion of a *committed UNC*.[7] This is a protocol for the two players A and B using a (γ, δ)-UNC and an error free channel. We shall assume that $\delta < 2\gamma(1 - \gamma)$, so that bit commitment can be done, based on the UNC. Note that if the UNC can only send bits from A to B, we can still simulate a UNC in the opposite direction using the error free channel as sketched in Subsection 2.9.3, so that we can assume that both A and B can commit to bits without loss of generality.

Intuitively, the purpose of a committed UNC is to act just like an ordinary UNC, but such that players are committed to the bits they send/receive on the UNC, at least except with some bounded probability.

We now define this concept more formally: a committed UNC protocol may halt because A or B reject. Otherwise it outputs two commitments, one from A containing a bit b_A, and one from B containing a bit b_B. Finally, the output designates one of the transmissions that were made over the UNC from A to B. Let s_A respectively r_B be the bit sent, respectively received in this transmission.

We require that if A, B both follow the protocol, then both players accept except with probability negligible in the security parameter k. Also, whenever A is honest, we have that b_A is uniformly random and $b_A = s_A$. Whenever B is honest, we have $r_B = b_B$. When A is corrupted and B is honest, we let p_A be the probability of the event that B accepts and $b_A \neq s_A$. Similarly, when B

[7] This technique originates from [Cré89].

is corrupted and A is honest, we let p_B be the probability that A accepts and $r_B \neq b_B$. In general, the error probabilities p_A, p_B will be functions of γ, δ and the security parameter k.

The argument sketched in [DKS99] on constructing OT from UNC took as point of departure a protocol that builds OT from a (γ, δ)-PassiveUNC for certain values of (γ, δ) and is secure assuming that players cheat only passively. It was then noted that one can replace the PassiveUNC with a UNC, still assuming that only passive cheating occurs. The final idea was then to replace the UNC with a committed UNC (although this notion was not formally defined there) and have players prove in ZK that they were following the protocol. If the error probabilities of the committed UNC could be made arbitrarily small with increasing k, then this would result in an OT secure against active cheating for essentially the same values of (γ, δ) that could be handled in the passive case. But unfortunately, this is impossible:

Theorem 6.1 *Any committed UNC as defined above, based on a (γ, δ)-UNC must have $p_A, p_B \geq \frac{\delta-\gamma}{1-2\gamma}$.*

Proof. Suppose, for instance, that A is cheating. Then A sets always the minimal noise level for the UNC, but adds artificial noise to each transmission with noise rate $\frac{\delta-\gamma}{1-2\gamma}$ such that the total error probability for each transmission is $\frac{\delta-\gamma}{1-2\gamma} \boxplus \gamma = \delta$. On the resulting transmissions, he runs a copy A_0 of the *honest* algorithm for A. Clearly, B (who is honest) cannot distinguish this from an all honest situation where the noise rate happens to be δ all the time, and so he must accept with overwhelming probability. However, it now holds for every transmission that the bit committed to and also sent by A_0, differs from the one A actually sent with probability $\frac{\delta-\gamma}{1-2\gamma}$. The theorem follows. □

Theorem 6.1 essentially says that we cannot force a player to commit to the bit he *physically* sends on a UNC. To get around this problem, we take a different point of view: we will create a new virtual channel from the UNC, where a bit committed to by the sender is *by definition* the bit sent on the new channel. Any difference between the committed bit and what is sent on the original UNC is regarded as noise. With appropriate checking that a cheating player does not introduce too much noise this way, it turns out that we obtain something that behaves essentially like a PassiveUNC, *even in presence of active cheating*. We model this by an ideal functionality called $(\gamma, \delta, q())$-Committed PassiveUNC (CPUNC). It combines a functionality similar to the PassiveUNC with the Commit-and-Prove functionality. In particular, it allows to commit to bits with or without sending them on the channel. But if they are sent, sender and receiver will be committed to what they send/receive. With security parameter k, the error rate will be in the range $\delta \pm 1/q(k)$, but will drop to γ given the view of a cheating player. Note that a CPUNC is not a committed UNC, and so Theorem 6.1 does not forbid the existence of a secure implementation.

Functionality $(\gamma, \delta, q())$-CPUNC

Stop: On receiving this command from the adversary, the CPUNC stops working and ignores all further commands.

Send cID, b**:** CPUNC comes with parameters $0 \leq \gamma \leq \delta \leq 1/2$, a security parameter value k and a polynomial $q()$. The issuer of the Send command is called the sender, the other party is the receiver. The string cID must not have been used before to identify a sent, received or committed bit, else the command is ignored. On receipt of this command from A or B, the functionality records cID, b and outputs a string "which error probability?" to the adversary, it ignores all further commands until the adversary sends an "Error probability" command.

Error probability κ'**:** Receiving this command from the adversary, the functionality checks if $|\delta - \kappa'| \leq 1/q(k)$. If not, the command is ignored. Otherwise, the functionality chooses random bits b', b'', such that $P[b' = 1] = \gamma$ and $P[b'' = 1] = \nu$, where $\nu \boxplus \gamma = \kappa'$.[8] The functionality sets $\hat{b} = b \oplus b' \oplus b''$. If the adversary has corrupted a player, it sends to the adversary a bit z, where $z = b \oplus b''$ if the sender is corrupted, and $z = b \oplus b'$ if the receiver is corrupted. It records cID, b as if the sender had committed to b. It then sends cID to both players, and ignores all further commands until the receiver sends a ReceiptID command.

ReceiptID $c\hat{I}D$**:** This command is ignored if $c\hat{I}D$ has been used to identify any committed, sent or received bit earlier. If this is not the case, the CPUNC records $c\hat{I}D, \hat{b}$ as if the receiver had committed to $\hat{b}$, it sends $c\hat{I}D$ to both players and $\hat{b}$ to the receiver.

Commit cID, b**:** Receiving this command, where cID is a bit-string and b is a bit, do as follows: if cID has not been used to identify a sent, received or committed bit before, record the value of cID, b and send as output $(Commit, cID)$ to both players.

Open cID**:** if cID, b has been recorded as a commitment from the player issuing this command, send b to both players.

Prove L, Φ**:** Receiving this command, where L is a list of bit strings and Φ is a Boolean formula, check if L contains only strings that has been used as identifiers for bits committed to by the issuer of the Prove command. If so, find the corresponding bits and check if they satisfy Φ. If so, sends (OK, L, Φ) to both players. Else, send $(Fail, L, \Phi)$.

We now describe a protocol that securely realizes the functionality we just described. We assume that the protocol has access to the UNC, CaP and RandomChoice functionalities. The protocol is described by specifying how each of the commands are implemented. The amount of work done in the protocol is specified by a polynomial $p(k)$, where k is the security parameter.

Protocol 6.7 CPUNC

Stop: This command has no direct implementation, the idea is that whenever the adversary behaves such that the honest party detects cheating and aborts, this is equivalent to sending a Stop command in the ideal scenario.

[8] This ensures that $P[b' \oplus b'' = 1] = \kappa'$.

Send b (Transmission Step): We describe how A will send a bit b to B.

1. A commits to b.
2. A chooses at random bits $\bar{B} = b_1, ..., b_{kp(k)^3}$, commits to each bit and sends each bit to B over the UNC. B commits to every bit $\hat{B} = \hat{b}_1, ..., \hat{b}_{kp(k)^3}$ he receives.
3. Call RandomChoice $kp(k)^2$ times to generate integers j_i chosen uniformly in the range $[1..kp(k)^3]$, for $i = 1, ..., kp(k)^2$.
4. All bits $b_{j_i}, \hat{b}_{j_i}$ are opened. Let κ be the fraction of the $kp(k)^2$ opened positions where $b_{j_i} \neq \hat{b}_{j_i}$. A and B check that $\kappa \leq \delta + 1/p(k)$. They abort all interaction if this is not satisfied.
5. Call RandomChoice to generate an integer j uniformly chosen among the indices of positions that were not opened in the previous step. A sends $b' = b \oplus b_j$ using error free transmission and proves (using CaP) that this value is correct.
6. Let μ be defined by $\kappa \boxplus \mu = \delta + 1/p(k)$. By a call to RandomChoice, generate a bit c such that $P[c = 1] = \mu$.
7. B defines the bit he receives as $\hat{b} = \hat{b_j} \oplus b' \oplus c$. He commits to $\hat{b}$ and proves (using CaP) that the committed value is correct.

If B wants to send a bit to A, we implement this in the same way as above, by interchanging the roles of A and B and of b_j and $\hat{b}_j$.

Commit, Open, Prove: Each of these commands correspond directly to commands that are already available in the Commit-and-Prove functionality we assume we have access to. Therefore these commands are implemented by directly calling the corresponding command with the same input in the Commit-and-Prove. Note that inputs to the Prove or Open command may include bits that were sent or received during a Send command, since these are also committed to.

Before proving anything about this construction, we describe first the intuition behind it: for bit-strings X, Y of equal length, let $err(X, Y)$ be the fraction of positions where X disagrees with Y. Now, if both parties are honest, the expected value of $err(\bar{B}, \hat{B})$ is at most δ, so allowing the estimate κ to be up to $\delta + 1/p(k)$ implies that we reject with negligible probability, as we shall see. Then assume that one player, say A, is corrupted, and let $\tilde{B} = \tilde{b}_1, ..., \tilde{b}_{kp(k)^3}$ be the bits actually sent by A on the UNC when a bit is transmitted. Let $\epsilon = err(\bar{B}, \tilde{B})$. Since the UNC introduces errors with probability γ independently of anything else, we expect that $\epsilon \boxplus \gamma \approx err(\bar{B}, \hat{B}) \approx \kappa$, and hence that $\epsilon \boxplus \gamma \boxplus \mu \approx \kappa \boxplus \mu \approx \delta$. Here, "$\approx$" means equality up to a $1/poly()$ term.

We can now see that after doing the transmission step, A is actually in a position approximately equivalent to having sent b on a (γ, δ)-PassiveUNC: we have that the bit b sent is related to the bit $\hat{b}$ received as $b = \hat{b} \oplus (b_j \oplus \tilde{b}_j) \oplus c \oplus n_j$, where n_j is a noise bit chosen by the UNC, such that $P[n_j = 1] = \gamma$. By the choice of c, and random choice of j, we have

$$P[b \neq \hat{b}] = P[(b_j \oplus \tilde{b}_j) \oplus c \oplus n_j = 1] \approx \epsilon \boxplus \mu \boxplus \gamma \approx \kappa \boxplus \mu \approx \delta.$$

But since the adversary knows $(b_j \oplus \tilde{b}_j) \oplus c$, the error rate from his point of view is only what is introduced by the UNC, namely γ.

We recall that a simulator (in the UC framework) is non-blocking, if it stops the CPUNC (by sending a stop command or refusing to give correct input when asked for it) with only negligible probability.

Theorem 6.2 *Protocol 6.7 securely realizes the* $(\gamma, \delta, q())$-CPUNC *functionality when given access to ideal* (γ, δ)-UNC, CaP *and* RandomChoice *functionalities, and for any polynomial* $q()$*, provided we choose the polynomial* $p()$ *measuring the work done in the protocol as* $p(k) = 4q(k)$*. Moreover, for the case where both players are honest, the simulator is non-blocking.*

Remark 6.3 We remind that the last claim in the theorem is a way to state in the UC framework the traditional completeness property for a two-party protocol: if both players are honest, the protocol completes successfully with overwhelming probability.

Proof. We begin with some technical lemmas. Assume that A or B is corrupted, let $\tilde{C}$ be the string actually sent or received by this player during a transmission step, let $\bar{C}$ be the string committed to by that player, and let $\hat{C}$ be the string the honest players sends or receives (and commits to). We can now describe the transmission step by the following equivalent random experiment: the adversary *Adv* chooses $\bar{C}, \tilde{C}$, and $\hat{C}$ is generated by sending $\tilde{C}$ through a binary symmetric channel with error probability γ. Then $kp(k)^2$ positions are chosen uniformly at random, $\bar{C}, \hat{C}$ are compared in these positions, and κ is the fraction of disagreements found. Finally, we define μ by $\kappa \boxplus \mu = \delta + 1/p(k)$.

We remind that a probability is called negligible in k if as a function of k, it converges to 0 faster than any polynomial fraction. Now, Lemma 2.1 trivially implies that κ is a good estimate of $err(\bar{C}, \hat{C})$:

Lemma 6.6 *For any given* $\bar{C}, \hat{C}$*,* $P[|\kappa - err(\bar{C}, \hat{C})| > 1/p(k)]$ *is negligible in* k*.*

Lemma 2.1 also implies:

Lemma 6.7 *For any given* $\bar{C}, \tilde{C}$*, let* $\epsilon = err(\bar{C}, \tilde{C})$*, and let* N *be the length of* $\bar{C}, \tilde{C}$ *(*$N = kp(k)^3$*). Then* $P[|err(\bar{C}, \hat{C}) - \epsilon \boxplus \gamma| > 1/p(k)]$ *is negligible in* k*.*

Proof. We substitute $n = N$ in Lemma 2.1, where the i'th experiment consists of flipping the i'th bit of $\tilde{C}$ with probability γ to get the i'th bit of $\hat{C}$. The event for each trial is that i'th bit of $\hat{C}$ turns out to be different from the i'th bit of $\bar{C}$. Then $\beta = err(\bar{C}, \hat{C})$, and the average of all the q_i's is easily seen to be $\epsilon \boxplus \gamma$. The lemma now follows immediately. □

The two previous lemmas immediately imply:

Lemma 6.8 *Except with negligible probability, any choice of* $\hat{C}, \tilde{C}$ *will lead to a value of* κ *(and hence* μ*) such that* $|\kappa - \epsilon \boxplus \gamma| \leq 2/p(k)$*, and hence by definition of* μ*, that* $|\delta + 1/p(k) - \epsilon \boxplus \gamma \boxplus \mu| \leq 2/p(k)$*.*

Since the number of positions we sample and compare between $\bar{C}, \hat{C}$ is much smaller than N, removing these positions does not change the expected error rates much. Concretely, let $\bar{C}_{uns}$ be the unsampled part of $\bar{C}$, and let $\tilde{C}_{uns}$ be the corresponding positions from $\tilde{C}$. We get:

Lemma 6.9 *Let $\epsilon' = err(\bar{C}_{uns}, \tilde{C}_{uns})$. Then $|\epsilon - \epsilon'| \leq 1/(p(k)-1)$ for all large enough k.*

Proof. Before we sample, $\bar{C}$ and $\tilde{C}$ disagree in ϵN positions. Sampling removes $s \leq kp(k)^2$ positions, so the number of disagreements for the unsampled part must be between ϵN and $\epsilon N - s$. It follows that

$$\epsilon + \frac{\epsilon}{N/s - 1} = \frac{\epsilon N}{N - s} \geq \epsilon' \geq \frac{\epsilon N - s}{N - s} = \epsilon - \frac{1 - \epsilon}{N/s - 1}$$

The lemma follows. □

Since $1/(p(k)-1) \leq 2/p(k)$ for all large enough k, combining the previous two lemmas immediately implies:

Lemma 6.10 *Except with negligible probability, any choice of $\hat{C}, \tilde{C}$ will lead to a value of κ (and hence of μ) such that $|\delta - \epsilon' \boxplus \gamma \boxplus \mu| \leq 4/p(k)$.*

The significance of this lemma is that, given the adversary's view up to the point where κ has just been determined, the probability that $b = \hat{b}$ is exactly $\epsilon' \boxplus \gamma \boxplus \mu$, and this will be important later.

We also need some technical lemmas for the case where both parties are honest:

Lemma 6.11 *If no player is corrupted, the transmission step will abort with only negligible probability.*

Proof. Clearly, the probability that the transmission step is aborted is maximal when the error rate of the UNC is always δ. In this case, the probability that one opened position shows disagreement is δ and this happens independently for each of the $kp(k)^2$ sampled positions, so we have $P[abort] \leq P[|\kappa - \delta| > 1/p(k)] \leq 2\exp(-k/2)$. □

Now, let $\alpha_1, ..., \alpha_{kp(k)^3}$ be a sequence of error probabilities chosen by the adversary for the UNC-transmissions in a transmission step as above, and consider the experiment where we run the protocol for A and B using the α_i's as error probabilities, until the point where κ and μ have been determined. Let α be the probability that $\hat{b}$ will be different from b, given the α_i's and the communication between A and B at this point, i.e., $\alpha = \mu \boxplus \bar{\alpha}$, where $\bar{\alpha}$ is the average of α_i's corresponding to unsampled positions.

Lemma 6.12 *Assume no player is corrupted. Starting from any (legal) sequence $\alpha_1, ..., \alpha_{kp(k)^3}$ as above, we obtain a value of α such that $|\delta - \alpha| \leq 4/p(k)$, except with negligible probability.*

Proof. Let $e(\alpha) = \sum_i \alpha_i / kp(k)^3$ be the average of all the α_i's. It follows immediately from Lemma 2.1 that $|\kappa - e(\alpha)| \leq 1/p(k)$ except with exponentially small probability. Now, let $\bar{\alpha}$ be the average of the α_i's taken only over unsampled positions. Since most positions are unsampled, clearly $\bar{\alpha}$ must be close to $e(\alpha)$. A straight forward calculation shows that in fact we always have $|\bar{\alpha} - e(\alpha)| \leq 2/(p(k) - 1)$. These two observations imply that $|\kappa - \bar{\alpha}| \leq 3/(p(k) - 1)$ except with negligible probability, which in turns implies that we also have $|\delta - \alpha| = |\kappa \boxplus \mu - \bar{\alpha} \boxplus \mu| \leq 3/(p(k) - 1)$, which is less than $4/p(k)$ for all large enough k. □

We are finally ready to show that we have a good implementation of the CPUNC functionality. To prove this, we need to construct a simulator S which will on one side interact with the adversary *Adv*. Since *Adv* expects to attack a scenario where the UNC, CaP and RandomChoice functionalities are available, S will simulate these in the natural way, for instance if *Adv* sends a bit on the UNC, S simply records this bit for later use. If the protocol calls for the received bit to become known to A, S adds a noise bit chosen by itself and gives the result to A. When RandomChoice is called, S just generates the output itself. As for calls to Commit, Open and Prove, S will record the bits sent, and then forward to the corresponding commands of the CPUNC. This is possible, since on the other side, S gets to attack an ideal scenario where we have the players A, B and the CPUNC. The goal is now to simulate *Adv*'s view of a real attack, as well as the results obtained by the honest player(s). We will only look at the cases where either A is corrupted or no one is corrupted. The case where where B is corrupted is similar and easily derived from the first case.

For the first: since *Adv* corrupts A, S will of course corrupted A in the ideal process. We then specify S by describing how it will react in each possible case where it is activated:

Send Command issued by A: When a send command is issued by *Adv* (who plays for A), the adversary must first commit to a bit b to send. S now goes through the protocol with *Adv* until κ, μ have been determined. If this leads to abort, S sends a Stop command to the CPUNC and halts. Otherwise S sends a Send b command to the CPUNC (on behalf of A). Then S looks at the unsampled bits from $\bar{B}$ and $\hat{B}$, and computes the fraction ϵ' of positions where they disagree. It sends an Error probability κ' command to the CPUNC, where $\kappa' = \epsilon' \boxplus \gamma \boxplus \mu$. It gets back a bit z.

S now chooses and sends to *Adv* the choices of j, c, to simulate the outputs of RandomChoice at this point. Of these, j is chosen among the indices of unsampled positions, either uniformly among indices where $b_j = \tilde{b}_j$, or among those where $b_j \neq \tilde{b}_j$ – we say that we choose $b_j \oplus \tilde{b}_j$ to be 0 or 1. Also c is chosen as 0 or 1. We choose among the 4 possible combinations as follows:

- If $z \oplus b = 0$, we let $b_j \oplus \tilde{b}_j = 0, c = 0$ with probability $(1-\epsilon')(1-\mu)/(1-\epsilon' \boxplus \mu)$, and we let $b_j \oplus \tilde{b}_j = 1, c = 1$ with probability $\epsilon'\mu/(1-\epsilon' \boxplus \mu)$.
- If $z \oplus b = 1$, we let $b_j \oplus \tilde{b}_j = 1, c = 0$ with probability $\epsilon'(1-\mu)/\epsilon' \boxplus \mu$, and we let $b_j \oplus \tilde{b}_j = 0, c = 1$ with probability $(1-\epsilon')\mu/\epsilon' \boxplus \mu$.

After this, S simply lets Adv finish the protocol (publishing $b' = b \oplus b_j$ and proving it relates in the right way to b and b_j). If Adv does not do this correctly, S will not deliver the output from the CPUNC to the receiver and will send a Stop command to the CPUNC.

Send command issued by B: S will learn that this command was issued when receiving a request for an error rate from the CPUNC. Having received this, S goes through the first part of the protocol for sending a bit with Adv, where Adv plays the role of the receiver. As above, this results in an error rate κ' being determined, where $\kappa' = \epsilon' \boxplus \gamma \boxplus \mu$, and where ϵ' is the error rate between the (unsampled) bits committed to by A and the bits actually received.

S sends κ' to CPUNC and gets a bit z from the CPUNC. Moreover, the CPUNC sends a bit $\hat{b}$ to the sender which is A, but since S corrupted A in the ideal process, the bit goes to S.

S now determines values j, c in the same way as above (where $\hat{b}$ replaces b), and sets $b' = \hat{b} \oplus \hat{b}_j$. It now tells the adversary that B sent b' and RandomChoice output b, j. Finally, it lets Adv complete the protocol, and sends a stop command to the CPUNC if this fails.

Commit, Open, Prove: Since these commands correspond to commands already available in the CaP, S can simulate any event relating to these commands by simply relaying input from the Adv to the CPUNC and output from the CPUNC back to Adv.

We now look at simulation in the second case, where no player is corrupted. In this case, all the adversary can do, is to select error probabilities each time a transmission takes place over the UNC (note that the argument that the adversary may as well select error probability γ always only holds in case a player is corrupted). What S does is therefore as follows:

Send Command issued by A: S learns that such a command has been issued when it receives a request for an error probability from the CPUNC. S then goes through the protocol with Adv which in this case just amounts to asking Adv for $kp(k)^3$ error probabilities $\alpha_1, ..., \alpha_{kp(k)^3}$ (which should all be between γ and δ).

Assuming Adv did this correctly, S selects an error probability α in the "right" way, given the α_i's. Concretely, S simulates (honestly) both A's and B's part of the protocol for sending a bit using the α_i's as error probabilities for the UNC transmissions. This simulation goes on until κ and μ have been determined. If $\kappa > \delta + 1/p(k)$, S sends a Stop command to the CPUNC and halts. Otherwise we compute an error probability $\alpha = \mu \boxplus \bar{\alpha}$, where $\bar{\alpha}$ is the average of the α_i's corresponding to unsampled bit positions. The value α is sent the CPUNC.

Send Command issued by B: - is handled in exactly the same way as if A issued the command.

Commit, Open, Prove: Since these commands correspond to commands already available in the CaP, S can simulate any event relating to these commands by simply relaying output from the CPUNC to Adv, typically cID's of bits committed to. However, there is no input from Adv to send to the CPUNC when no one is corrupted.

We sketch a proof that this simulation is good. We first look at the case where A is corrupted:

Send Command issued by A: We can first remark that the simulation of A's view is perfect until the point where κ, μ are determined. In particular, by Lemma 6.10 we have $|\delta - \epsilon' \boxplus \gamma \boxplus \mu| < 1/q(k)$, except with negligible probability (recall that we let $p(k) = 4q(k)$ in the claim), so we may assume in the following that this holds. Therefore, the κ' that S asks the CPUNC to use will be close enough to δ for the CPUNC to accept this. Note that if A fails to complete the protocol for the Send command, no output is generated and no further interaction takes place. This happens both in simulation and in real life. So we now argue, assuming that A completes the protocol. Note first that, given Adv's view, the probability that $b = \hat{b}$ is exactly $\epsilon' \boxplus \gamma \boxplus \mu$ in both simulation and in real life, so the honest player receives a correctly distributed bit. Moreover, it follows directly from the algorithm of CPUNC that $P[z \neq b] = \epsilon' \boxplus \mu$. Therefore elementary probability calculations show that j will be uniformly distributed over the unsampled positions, and c will be independent, and be 1 with probability μ. So this matches the distribution of the view of A in real life. Furthermore, in real life, the correlation between A's view and the bit received by B is completely described by $P[\tilde{b}_j \oplus b' \oplus c \neq \hat{b}] = \gamma$. But this is also the case in the simulation: the algorithm of S ensures that $z \oplus b = b_j \oplus \tilde{b}_j \oplus c$, and A must prove that $b = b_j \oplus b'$. Combining these two, we get $z = b' \oplus \tilde{b}_j \oplus c$, and by definition of the CPUNC, $P[z \neq \hat{b}] = \gamma$.

Send command Issued by B: The argument for this case is easily derived from the above case.

Commit,Open,Prove: The simulation of these events is trivially perfect, since the CPUNC by definition behaves exactly like CaP on any of these commands.

We finally show that the simulation is good in the case where both players are honest:

Send Command issued by A: The simulation is clearly perfect, except for the cases where the α sent by S is further away than $1/q(k)$ from δ. In these cases, the real-life protocol may complete, whereas the CPUNC would always block the transmission. However, by Lemma 6.12, this only happens with negligible probability. Furthermore, the simulator only blocks the CPUNC if the value of κ is too large or α is illegal. By Lemmas 6.11 and 6.12, this only happens with negligible probability, so the simulator is non-blocking in this case.

Send command Issued by B: The argument for this case is the same as for the above case.

Commit,Open,Prove: The simulation of these events is trivially perfect, since the CPUNC by definition behaves exactly like CaP on any of these commands.

□

6.4.2 From Passive to Active Security

In this subsection, we sketch a proof of the following result:

Theorem 6.3 *Let π be any protocol that securely realizes OT based on a (γ, δ)-PassiveUNC assuming a passive adversary. Then there exists a protocol with complexity polynomial in that of π that also securely realizes OT based on a (γ, δ)-UNC, assuming an active adversary.*

Proof. We assume we have a protocol π that implements Oblivious Transfer given access to a (γ, δ)-PassiveUNC functionality, and that this protocol is secure against a passive adversary.

We then note that the previous subsection showed how to implement the CPUNC functionality based on the UNC. Therefore from π, we may construct a protocol $\bar{\pi}$ as follows: active cheating is prevented by first making players commit to all inputs, and furthermore, the random coins of a player are decided using a standard technique: the player in question commits to a random string a, the other player sends a random string b in the clear and the random coins to be used are $a \oplus b$. Second, all transmissions over the PassiveUNC now take place using the CPUNC, and each time something is sent, one uses the CPUNC to prove that what was sent was computed according to π with the given (committed) inputs, random coins and messages received earlier.

Note that a player trying to send an incorrect message will be caught with certainty. Therefore, the views obtained by the players are always (a possibly truncated version of) what would be obtained in presence of a passive adversary.

Our first goal will be to show that $\bar{\pi}$ implements a (p, q, ϵ)-WOT as defined in Subsection 6.2.3.

Lemma 6.13 *$\bar{\pi}$ as described above realizes (with statistically good simulation) a (p, q, ϵ)-WOT with $p = q = \epsilon = 3/k$, when $\bar{\pi}$ is executed with security parameter value k.*

Proof. (Sketch) The above discussion implies that we only have to show the lemma for a passive adversary: the only difference between a passive and an active attack on $\bar{\pi}$ is that the adversary may stop early in the active case, and this can never be prevented in an active attack. Assuming a passive adversary, the only difference between $\bar{\pi}$ and π is that $\bar{\pi}$ does not use a (γ, δ)-PassiveUNC but a $(\gamma, \delta, f())$-CPUNC where the adversary can make the error probability fluctuate slightly around δ. This fluctuation is not negligible, namely it is of

size $1/f(k)$. However, by Theorem 6.2, we can choose $f()$ to be any polynomial we like, so assuming π calls the PassiveUNC $t(k)$ times, for some polynomial $t()$, we choose $f(k) = kt(k)$.

Consider the view of a (passively) corrupted sender in π, represented by random variable V. Let $adv_\pi(k, v)$ be the advantage over 1/2 with which the selection bit can be guessed given that $V = v$ and the protocol was executed with security parameter value k. Let $adv_\pi(k) = \sum_v P[V = v] \cdot adv_\pi(k, v)$ be the expected value. Since π was assumed to be secure, $adv_\pi(k)$ is negligible in k (this is equivalent to asserting that the mutual information between the selection bit and V is negligible). Then define a particular possible value v of V to be *good* if $adv_\pi(k, v) \leq \sqrt{adv_\pi(k)}$, and let $\mathcal{E}$ be the event that V takes a bad value. Then clearly, $\mathcal{E}$ occurs with probability at most $\sqrt{adv_\pi(k)}$. We now define $t(k) + 1$ hybrids that are in between π and $\bar{\pi}$: namely in the i'th hybrid, where $i = 0, \ldots, t(k)$, we run the normal protocol, but for communication, we use a (γ, δ)-PassiveUNC for the first i calls to the communication channel, and then the $(\gamma, \delta, q())$-CPUNC for the rest. Then hybrid 0 is $\bar{\pi}$ while hybrid $t(k)$ is π. When executing hybrid i, we define $\mathcal{E}_i$ to be the event that the information contained in the sender's view about the selection bit is larger than $\sqrt{adv_\pi(k)}$. Let ϵ_i be the probability that $\mathcal{E}_i$ occurs. Of course $\epsilon_{t(k)} = P[\mathcal{E}] \leq \sqrt{adv_\pi(k)}$. Also, the only difference between hybrid i and $i + 1$ is that in the $i + 1$'st call to the communication channel, the results returned by the channel have distributions with statistical difference at most $2/f(k)$ between them. It follows that $|\epsilon_i - \epsilon_{i+1}| \leq 2/f(k)$, and hence $\epsilon_0 \leq \epsilon_{t(k)} + 2t(k)/f(k) \leq \sqrt{adv_\pi(k)} + 2/k$. The "OT", that $\bar{\pi}$ implements is therefore no worse than a protocol that with probability, say $3/k$ reveals the selection bit to the sender, and otherwise leaks a negligible amount of information. A similar argument holds for the view of a corrupted receiver; also this type of argument shows that an honest receiver will receive the correct bit, except with probability at most $3/k$. Thus what we have is statistically indistinguishable from a (p, q, ϵ)-WOT, with $p = q = \epsilon = 3/k$. □

We can then complete the argument for the theorem: taking into account Lemma 6.5, OT can be implemented based on any (p, q, ϵ)-WOT, as long as $p + q + 2\epsilon < 0.45$. Moreover, it is easy to verify that by choosing k large enough the reduction implements OT efficiently, i.e., it only makes a polynomial number of calls to the underlying WOT. Therefore, by Lemma 6.13, we can replace the WOT by $\bar{\pi}$ and still obtain a secure OT (even though $\bar{\pi}$ is only statistically close to the required WOT). This implies the result we wanted. □

6.5 Extended Possibility Results

In this section, we shall assume the result of Theorem 6.3 and focus on reducing OT to (γ, δ)-PassiveUNC secure against passive adversaries.

The first observation leading towards the new possibility result is that WOT does not precisely capture the imperfect OT produced by Protocol 6.3: In WOT the corrupted sender/receiver gets the selection/secret bit (which he is not supposed to see) with a certain probability, while in the imperfect OT

obtained the corrupted sender/receiver only gets *some information* about that bit with a certain probability: by inspection of Protocol 6.3, it is clear that the corrupted player never gets the private input of the other party "for free", i.e., with certainty.

As a consequence, the information leakage to the adversary is overestimated in [DKS99], the fact already known to the authors. We introduce a new *Generalised Weak Oblivious Transfer (GWOT)* primitive which allows to model imperfect OT's which leak information about the parties' private inputs in a much more general way than WOT's, without overestimating the information leakage. In particular, it precisely captures the imperfect OT produced by Protocol 6.3. Informally, in a GWOT the corrupted sender/receiver gets the selection/secret bit over a BSC with some error probability which is chosen according to some distribution (and announced to the corrupted party). Formally, consider parameters $\{s_i, \alpha_i\}_i$ and $\{r_i, \beta_i\}_i$, where $i = 1, \ldots, N$, and ϵ such that $\{s_i\}_i$ and $\{r_i\}_i$ are probability distributions (over $\{1, \ldots, N\}$) and $0 \leq \alpha_i, \beta_i, \epsilon \leq 1/2$ for $i = 1, \ldots, N$. A GWOT with respect to these parameters is specified by a primitive of the following kind.

Definition 6.1 *$(\{(s_i, \alpha_i)\}_{i=1}^N; \{(r_i, \beta_i)\}_{i=1}^N; \epsilon)$-GWOT is an OT with the following faults:*

- *A learns the selection bit c in the following way: the values $I \in \{1, \ldots, N\}$ and $\tilde{c} \in \{0, 1\}$ are drawn at random such that $P[I = i] = s_i$ and $P[\tilde{c} \neq c | I = i] = \alpha_i$ and then A learns I and $\tilde{c}$.*
- *B learns the secret bit b_{1-c} in the following way: the values $I \in \{1, \ldots, N\}$ and $\tilde{b}_{1-c} \in \{0, 1\}$ are drawn at random such that $P[I = i] = r_i$ and $P[\tilde{b}_{1-c} \neq b_{1-c} | I = i] = \beta_i$ and then B learns I and b_{1-c}.*
- *with probability ϵ the honest receiver B gets $1 - b_c$ instead of b_c (i.e., incorrect value).*

We will say that a corrupted sender gets c "sent through $\{(s_i, \alpha_i)\}_{i=1}^N$" and similarly a corrupted receiver gets b_{1-c} "sent through $\{(r_i, \beta_i)\}_{i=1}^N$".

Note that there is some ambiguity in the primitive's action in that it is not required that $\tilde{b}_c$ is chosen independently of I and $\tilde{c}$, respectively of I and $\tilde{b}_{1-c}$, as long as the marginal distribution of $\tilde{b}_c$ is correct. Furthermore, a (p, q, ϵ)-WOT coincides obviously with a $\big(\{(p, 0), (1-p, 1/2)\}; \{(q, 0), (1-q, 1/2)\}; \epsilon\big)$-GWOT.

It will be convenient to introduce a GWOT of a very particular form, a *Special Generalised Weak Oblivious Transfer* (*SGWOT*). Informally, in a SGWOT the corrupted sender/receiver either gets no information on the selection/secret bit or he receives it over a BSC with a certain (fixed) error probability. Formally, for parameters $s, \alpha, r, \beta, \epsilon$ with $0 \leq s, r \leq 1$ and $0 \leq \alpha, \beta \leq 1/2$,

$$\big((s, \alpha), (r, \beta), \epsilon\big)\text{-SGWOT} \stackrel{def}{=} \big(\{(s, 1/2), (1-s, \alpha)\}; \{(r, 1/2), (1-r, \beta)\}; \epsilon\big)\text{-GWOT}.$$

Let us consider Protocol 6.3. As mentioned above, this construction actually results in a GWOT (which is modelled by a WOT by giving away information

to the adversary). As a matter of fact, as it can easily be seen, it results in a SGWOT. The following lemma expresses the parameters of the resulting SGWOT as a function of (γ, δ).

Let μ be such that $\delta = \mu \boxplus \gamma$. The proof of the lemma follows by straightforward case analysis of Protocol 6.3.

Lemma 6.14 *When run with a (γ, δ)-PassiveUNC, Protocol 6.3 produces a $((s, \alpha), (r, \beta), \epsilon)$-SGWOT with the following parameters:*

$$s = \frac{\gamma(1-\gamma)(\gamma^2 + (1-\gamma)^2)(\mu^4 + 6\mu^2(1-\mu)^2 + (1-\mu)^4)}{\delta(1-\delta)(\delta^2 + (1-\delta)^2)},$$

$$\alpha = \frac{4\gamma^2(1-\gamma)^2}{\gamma^4 + 6\gamma^2(1-\gamma^2) + (1-\gamma^4)}, \quad (6.1)$$

$$r = \frac{\gamma(1-\gamma)(\mu^2 + (1-\mu)^2)}{\delta(1-\delta)}, \quad \beta = \frac{\gamma^2}{\gamma^2 + (1-\gamma)^2} \quad (6.2)$$

$$\epsilon = \frac{\delta^2}{\delta^2 + (1-\delta)^2}. \quad (6.3)$$

We have expressed the parameters of $((s, \alpha), (r, \beta), \epsilon)$-SGWOT with that of the underlying (γ, δ)-PassiveUNC. Now, we shall exploit the machinery of [DKS99] in order to reduce OT to SGWOT.

In [DKS99], the $((s, \alpha), (r, \beta), \epsilon)$-SGWOT obtained after invoking Reduction 6.3 was modelled by a $(1-s, 1-r, \epsilon)$-WOT. I.e., in order to fit the imperfect OT into the WOT framework, the error probabilities α and β were assumed to be zero by giving the corrupted party some information for free. Clearly, a tighter analysis should avoid this kind of strengthening of the corrupted party for proof-technical conveniences. A straight forward approach would be to try to show that for certain initial parameters, the sequence of GWOT's produced by applying the S-, R- and E-Red reductions to the initial SGWOT, converges to an OT. Unfortunately, as the reduction of OT to WOT defined in [DKS99] is executed, the shape of the GWOT's becomes quickly very complex and difficult to analyse. In order to avoid this problem, we give a generic way to replace a (possibly very complex) GWOT by another (ideally simpler) one such that if the new GWOT allows for OT then the initial GWOT also allows for OT; however, in contrast to the strategy of [DKS99] of simply setting the error probabilities to zero, we are trying to be much more tight.

Next definition introduces a partial ordering "$\preceq$" among probability distributions over BSC's, i.e., among sets of the form $\{(s_i, \alpha_i)\}_i$ or $\{(r_i, \beta_i)\}_i$ as considered above, that will be shown (in Lemma 6.15) to capture the relative difficulty to generate OT using the reduction considered in [DKS99]. Intuitively, we say that $S \preceq S'$ if S can be transformed into S' by removing BSC's in S and replacing each of them by a Bernoulli distribution over two BSC's such that the *average* guessing probability for the bit sent through S is the same as when sent through S'.

Definition 6.2 *Let $S = \{(p_i, \epsilon_i)\}_{i=1}^N$ and S' be two probability distributions over BSC's. We say that $S \preceq S'$ if there exists $1 \leq \ell \leq N$ as well as $0 \leq \delta \leq 1$ and $0 \leq \epsilon^- \leq \epsilon \leq \epsilon^+ \leq 1/2$ such that*

1. *S' is of the form $S' = S \setminus \{(p_\ell, \epsilon_\ell)\} \cup \{((1-\delta)p_\ell, \epsilon^-), (\delta p_\ell, \epsilon^+)\}$ and*
2. *$\epsilon_\ell = \epsilon = (1-\delta) \cdot \epsilon^- + \delta \cdot \epsilon^+$,*

or if there exists a sequence $S = S_0, S_1, \ldots, S_k = S'$ of probability distributions over BSC's such that $S_{\kappa-1} \preceq S_\kappa$ in the above sense for $\kappa = 1, \ldots, k$.

Note that in case $\epsilon_j = \epsilon_k$ for some $1 \leq j < k \leq N$, we identify $S = \{(p_i, \epsilon_i)\}_{i=1}^N$ with $S^* = S \setminus \{(p_j, \epsilon_j), (p_k, \epsilon_k)\} \cup \{(p_j + p_k, \epsilon_j)\}$. This is justified in that it is immaterial in our context whether a bit is sent through S or through S^*.

The next lemma proved by Salvail and Fehr shows that the partial ordering $S \preceq S'$ means that as long as reductions S-Red, R-Red, and E-Red are concerned, S is *easier* to deal with than S'.

Lemma 6.15 *If OT can be reduced to $(S'; R'; \epsilon)$-GWOT by a sequence of reductions S-Red, R-Red, and E-Red, then OT can be reduced to any $(S; R; \epsilon)$-GWOT with $S \preceq S'$ and $R \preceq R'$.*

Proof. In the following we show that S can be replaced by S' and R can be replaced by R' to the advantage of the adversary for any sequence of the basic reductions S-Red(l), R-Red(l), and E-Red(l). We write $P = S$, $P' = S'$ and $b = c$ for the selection bit c in case the sender is corrupted, and $P = R$, $P' = R'$ and $b = b_{1-c}$ for the secret bit b_{1-c} in case the receiver is corrupted. $P \preceq P'$ implies that there exists a sequence of transformations $P = P_0, P_1, \ldots, P_k = P'$ where each P_κ is obtained from $P_{\kappa-1}$ as described in Definition 6.2. We show that the (expected) guessing probability of the corrupted sender respectively receiver for the bit b does not decrease when $P_{\kappa-1}$ is replaced by P_κ in the description of the GWOT. In the following we fix $0 < \kappa \leq k$ and assume $P_{\kappa-1} = \{(p_i, \epsilon_i)\}_{i=1}^N$. Let $0 < \ell \leq N$, $0 \leq \delta \leq 1$ and $0 \leq \epsilon^- \leq \epsilon \leq \epsilon^+ \leq 1/2$, be defined as in Definition 6.2.

We fix some notation. Consider the transmission of l bits $x_1, \ldots, x_l$ which encode b as specified later through $P_{\kappa-1}$. Formally, for $j = 1, \ldots, l$, a channel (index) $I_j \in \{1, \ldots, N\}$ is independently chosen such that $P[I_j = i] = p_i$ and a bit y_j such that it differs from x_j with probability ϵ_{I_j}. Write $I = [I_1, \ldots, I_l]$ and $y = [y_1, \ldots, y_l]$. Finally, let $I' = [I'_1, \ldots, I'_l]$ and $y' = [y'_1, \ldots, y'_l]$ be such that $I'_j = I_j$ and $y'_j = y_j$ if $I_j \neq \ell$ and otherwise I'_j is chosen from $\{+, -\}$ such that $P[I'_j = +] = \delta$ and $y'_j \in \{0, 1\}$ such that it differs from x_j with probability $\epsilon_{I'_j}$. Hence, I' and y' can be viewed as the outcome of sending $x_1, \ldots, x_l$ through P_κ. Let *guess* be the probability of guessing (random) b correctly given I and y (using an optimal guessing strategy), and similarly let *guess*$'$ be the guessing probability for b given I' and y'. We will show that *guess* $\geq$ *guess*$'$, i.e., replacing $P_{\kappa-1}$ by P_κ only helps in guessing b.

For simplicity, we assume that there exists exactly one j such that $I_j = \ell$. If there is none then the claim definitely holds, and the case of several I_j's being equal to ℓ can be reduced to the case of one using a straight forward hybrid argument. Let j^* be that special j. Write v for the collection of the I_j's and y_j's (or, equivalently, I'_j's and y'_j's) with $j \neq j^*$, and write c for y_{j^*} as well as c' for (I'_{j^*}, y'_{j^*}). In the following we assume an arbitrary but fixed value for v (which

has non-zero probability), and we show that $guess \geq guess'$ for that v. This of course implies that $guess \geq guess'$ for v chosen according to its distribution. Formally, in the following analysis, we consider the probability space obtained by conditioning on the event that v takes on the considered value.

There are two distinct cases to analyse. The first case is when the bit b was split into l parts $(x_1, \ldots, x_l)$ such that $\bigoplus_j x_j = b$. This corresponds to the situation where the adversary is the sender in S-Red(l) or the receiver in R-Red(l). Consider the optimal guess for $b \oplus x_{j^*} = \bigoplus_{j \neq j^*} x_j$. If this guess is correct, then $guess$ and $guess'$ are given by the guessing probabilities for x_{j^*} given c respectively c'. If it is incorrect, then $guess$ and $guess'$ are given by the probabilities of guessing x_{j^*} wrongly given c respectively c'. In either case, these probabilities coincide by the assumption on δ, ϵ, ϵ^+ and ϵ^- posed in Definition 6.2, and hence $guess = guess'$.

The second case is when b is *sent* l times through $P_{\kappa-1}$ respectively P_κ, i.e., $x_1 = \cdots = x_l = b$. This corresponds to the situation where the adversary is the sender in R-Red(l) or the receiver in S-Red(l) or the adversary is either the receiver or the sender in E-Red(l). Let ρ_b be the probability of observing v given b (in the original unconditioned probability space), and write $\alpha = \rho_1/\rho_0$. Also, let $G(c)$ denote the guessing probability for b depending on c. Then

$$\begin{aligned} G(0) &= \max\left(\frac{\rho_0(1-\epsilon)}{\rho_0(1-\epsilon)+\rho_1\epsilon}, \frac{\rho_1\epsilon}{\rho_0(1-\epsilon)+\rho_1\epsilon}\right) = \frac{\max(1-\epsilon, \alpha\epsilon)}{(1-\epsilon)+\alpha\epsilon} \quad \text{and} \\ G(1) &= \max\left(\frac{\rho_0\epsilon}{\rho_0\epsilon+\rho_1(1-\epsilon)}, \frac{\rho_1(1-\epsilon)}{\rho_0(1-\epsilon)+\rho_1\epsilon}\right) = \frac{\max(\epsilon, \alpha(1-\epsilon))}{\epsilon+\alpha(1-\epsilon)}\,. \end{aligned}$$

In both cases, the two terms in the max are the success probabilities when guessing b to be 0 and 1, respectively. The probabilities that $c = 0$ and that $c = 1$ are given by

$$\begin{aligned} P[c=0] &= \frac{\rho_0(1-\epsilon)+\rho_1\epsilon}{\rho_0+\rho_1} = \frac{(1-\epsilon)+\alpha\epsilon}{1+\alpha} \quad \text{and} \\ P[c=1] &= \frac{\rho_0\epsilon+\rho_1(1-\epsilon)}{\rho_0+\rho_1} = \frac{\epsilon+\alpha(1-\epsilon)}{1+\alpha}\,. \end{aligned}$$

Therefore,

$$\begin{aligned} guess &= \frac{(1-\epsilon)+\alpha\epsilon}{1+\alpha} \cdot \frac{\max(1-\epsilon, \alpha\epsilon)}{(1-\epsilon)+\alpha\epsilon} + \frac{\epsilon+\alpha(1-\epsilon)}{1+\alpha} \cdot \frac{\max(\epsilon, \alpha(1-\epsilon))}{\epsilon+\alpha(1-\epsilon)} \\ &= \frac{\max(1-\epsilon, \alpha\epsilon)}{1+\alpha} + \frac{\max(\epsilon, \alpha(1-\epsilon))}{1+\alpha} \end{aligned}$$

Since this expression is invariant under replacing α by $1/\alpha$, we may assume that $0 \leq \alpha \leq 1$, and hence

$$guess = \frac{1}{1+\alpha} \cdot \big(\alpha\epsilon + \max(\epsilon, \alpha(1-\epsilon))\big)\,.$$

Similarly, it holds that

$$guess' = \frac{\delta}{1+\alpha} \cdot \big(\alpha\epsilon^+ + \max(\epsilon^+, \alpha(1-\epsilon^+))\big) + \frac{1-\delta}{1+\alpha} \cdot \big(\alpha\epsilon^- + \max(\epsilon^-, \alpha(1-\epsilon^-))\big)\,.$$

Therefore,
$guess' =$

$$\begin{aligned}
&= \frac{1}{1+\alpha} \cdot \big(\alpha\epsilon^+ + \delta \cdot \max(\epsilon^+, \alpha(1-\epsilon^+)) + (1-\delta) \cdot \max(\epsilon^-, \alpha(1-\epsilon^-))\big) \\
&= \frac{1}{1+\alpha} \cdot \big(\alpha\epsilon^+ + \max(\delta\epsilon^+, \delta\alpha(1-\epsilon^+)) + \max((1-\delta)\epsilon^-, (1-\delta)\alpha(1-\epsilon^-))\big) \\
&\geq \frac{1}{1+\alpha} \cdot \big(\alpha\epsilon^+ + \max(\delta\epsilon^+ + (1-\delta)\epsilon^-, \delta\alpha(1-\epsilon^+) + (1-\delta)\alpha(1-\epsilon^-))\big) \\
&= \frac{1}{1+\alpha} \cdot \big(\alpha\epsilon + \max(\epsilon, \alpha(1-\epsilon))\big) \\
&= guess
\end{aligned}$$

This had to be shown. □

In particular, Lemma 6.15 allows to improve the analysis of [DKS99]. As we have seen in Lemma 6.14, the imperfect OT obtained from a PassiveUNC using Reduction 6.3 produces a $((s,\alpha),(r,\beta),\epsilon)$-SGWOT. Using Lemma 6.15, it is straight forward to verify that we can replace this SGWOT by a (p_s, q_r, ϵ)-WOT with $p_s = (1-s)(1-2\alpha)$ and $q_r = (1-r)(1-2\beta)$. Indeed, for instance the corrupted sender's guessing probability for the selection bit is in the first case $s/2 + (1-s)(1-\alpha) = 1 - s/2 - \alpha + s\alpha$ and in the second case $p_s + (1-p_s)/2 = 1 - s/2 - \alpha + s\alpha$. Applying Lemma 6.5 to the transformed SGWOT results in the following lemma.

Lemma 6.16 *There exists a reduction of OT to $((s,\alpha),(r,\beta),\epsilon)$-SGWOT with $p_s + q_r + 2\epsilon \leq 0.45$ where $p_s = (1-s)(1-2\alpha)$ and $q_r = (1-r)(1-2\beta)$.*

Combining Lemmas 6.14 and 6.16 gives directly the following result:

Lemma 6.17 *OT can be reduced to (γ,δ)-PassiveUNC if $p_s + q_r + 2\epsilon \leq 0.45$, where $p_s = (1-s)(1-2\alpha)$, $q_r = (1-r)(1-2\beta)$ and s, α, r, β, ϵ are defined by equations (6.1)–(6.3).*

Note that in Lemma 6.5, OT can only be achieved if $p+q+2\epsilon \leq 0.45$ where $p = 1-s$ and $q = 1-r$. Hence, the possibility range of (γ,δ) values given in Lemma 6.17 strictly contains the one obtained in [DKS99]. This can be seen from Figure 6.1, where these possibility results are shown. We note that the thin diagonal line $\gamma = \delta$ on Figure 6.1 depicts the Binary Symmetric Channel. We are investigating the range below it, since this range represents the UNC's with $\gamma < \delta$. The area below the curve $\delta = 2\gamma(1-\gamma)$ (we shall call this curve the *simulation bound*) is the range where no primitives are possible, since the UNC's are trivial there according to Lemmas 6.1 and 6.2. As mentioned above, bit commitments are possible everywhere between the simulation bound and the BSC line. Our goal is to get the possibility result for OT which is as close as possible to the simulation bound.

Despite some improvement, Lemma 6.17 still shares the following restriction with [DKS99]: OT is not known to be possible for $\delta > 0.35$ even if γ is almost equal to δ (i.e., the corresponding UNC is "almost fair") since in that case

$\epsilon > 0.45$ and so the condition of Lemma 6.5 is not satisfied (see Figure 6.1). This stands somewhat in contrast to the fact that OT can be achieved based on any (non-trivial) BSC as it is shown in Chapter 3. Hence, one might expect that OT can be achieved based on any (non-trivial) UNC as long as the unfairness is small enough. The following lemma shows that this intution is indeed true.

Lemma 6.18 *There exists a reduction of OT to any (γ, δ)-PassiveUNC that satisfies $1-(1-p_s)^l + 1-(1-q_r)^l + 2\frac{\epsilon^l}{\epsilon^l+(1-\epsilon)^l} \leq 0.45$ for some $l \geq 1$, where $p_s = (1-s)(1-2\alpha)$ and $q_r = (1-r)(1-2\beta)$ with $s, \alpha, r, \beta, \epsilon$ defined by (6.1)–(6.3).*

Remark 6.4 Clearly, for any $0 < \delta < 1/2$, for l large enough, and for γ close enough to δ (where the closer δ is to $1/2$, the closer γ has to be to δ), the values p_s and q_r are small enough for the condition expressed in Lemma 6.18 to be satisfied. Hence, OT is possible based on (γ, δ)-PassiveUNC's for any $0 < \delta < 1/2$ as long as γ is close enough to δ (see Figure 6.1). This further improves on the results of [DKS99].

Proof. We implement a $((s,\alpha),(r,\beta),\epsilon)$-SGWOT from the (γ,δ)-PassiveUNC according to Lemma 6.14. Using Lemma 6.15, we convert it into a (p_s, q_r, ϵ)-WOT and then apply the reduction E-Red(l). By Lemma 6.4, this results in a $\left(1-(1-p_s)^l, 1-(1-q_r)^l, \frac{\epsilon^l}{\epsilon^l+(1-\epsilon)^l}\right)$-WOT. The claim now follows. □

It can be shown by straight forward calculations that the new possibility range includes UNC's for which the techniques of [DKS99] results in the trivial WOT's, i.e., the ones that could not be used to implement OT (see Lemma 6.3). In other words, our approach allows to implement and prove secure OT in a range where it is *provably impossible* using the techniques of [DKS99]. The following example illustrates this.

Example 6.1 Let $\gamma_0 = 0.39$, $\delta_0 = 0.4$ be the parameters of a PassiveUNC. The $(p(\gamma_0,\delta_0), q(\gamma_0,\delta_0), \epsilon(\delta_0))$-WOT obtained from a (γ_0,δ_0)-PassiveUNC the crude way (by giving away all partial information to the adversary as in [DKS99]) achieves $p(\gamma_0,\delta_0) + q(\gamma_0,\delta_0) + 2\epsilon(\delta_0) \approx 0.869$. It can be shown that from this WOT, any sequence of reductions S-, R- and E-Red generates a simulatable WOT, i.e., OT is not reducible to the $(p(\gamma_0,\delta_0), q(\gamma_0,\delta_0), \epsilon(\delta_0))$-WOT using S-, R- and E-Red. At the same time, the $(p_s(\gamma_0,\delta_0), q_r(\gamma_0,\delta_0), \epsilon(\delta_0)$-WOT (obtained according Lemma 6.15) achieves $p_s(\gamma_0,\delta_0) + q_r(\gamma_0,\delta_0) + 2\epsilon(\delta_0) \approx 0.671$. Moreover, E-Red(2) applied to this WOT generates a (p', q', ϵ')-WOT with $p' + q' + 2\epsilon' \approx 0.438$, which we know from Lemma 6.17 implies OT.

There exists an even larger range than the one described in Lemma 6.18 for which a possibility result can be shown. This follows from the fact that the approach of Lemma 6.18 still gives information for free to the adversary. Indeed, the SGWOT obtained from a (γ,δ)-PassiveUNC is converted into a (p_s, q_r, ϵ)-WOT before reductions S-Red, R-Red and E-Red are applied. We may benefit from trying to preserve the SGWOT through the sequence of reductions.

The problem is that the reductions do not preserve the SGWOT per se but produce more complex GWOT's with a quickly growing set of parameters. An approach is to use Lemma 6.15 in order to immediately convert any resulting GWOT (which is not a SGWOT) back into a SGWOT. Specifically, a $(\{(s_i,\alpha_i)\}_i;\{(r_i,\beta_i)\}_i;\epsilon)$-GWOT can be replaced by a $((s,\alpha),(r,\beta),\epsilon)$-SGWOT, where $\alpha=\min_i \alpha_i$ and $\beta=\min_i \beta_i$, and s and r are appropriately chosen such that $\{(s_i,\alpha_i)\}_i \preceq \{(s,1/2),(1-s,\alpha)\}$ and $\{(r_i,\beta_i)\}_i \preceq \{(r,1/2),(1-r,\beta)\}$. This indeed results in an increased possibility range.

Lemma 6.19 *There exists a range of values (γ,δ) which do not satisfy the conditions of Lemma 6.18 but where OT can still be implemented from such (γ,δ)-PassiveUNC's.*

Proof. (sketch) By brute force analysis for any fixed value of δ_0, $0<\delta_0<1/2$, we find the smallest value of γ_0, such that a SGWOT based on (γ_0,δ_0)-PassiveUNC can be reduced to a SGWOT with $p_s+q_r+2\epsilon\leq 0.45$ using the reductions S-Red, R-Red and E-Red, and replacing any GWOT by a SGWOT as sketched above.

For example, let $\gamma_0=0.365$, $\delta_0=0.4$. The value $p_s+q_r+2\epsilon$ of the SGWOT resulting from (γ_0,δ_0)-PassiveUNC is equal to 0.793. It is easy to check that the conditions of Lemma 6.18 are not satisfied with respect to this SGWOT. Nonetheless, the sequence of reductions "EERSRESERRSESRERSESERRS" (i.e., E-Red(2), then E-Red(2) again, then S-Red(2) and so on; and each reduction has parameter $k_S=k_R=k_E=2$) produces as output a SGWOT with $p_s+q_r+2\epsilon=0.329$ which implies OT according to Lemma 6.17. □

Using brute-force analysis, it is possible to find numerically the range for which the reduction considered in Lemma 6.19 produces OT. The new range is depicted on Figure 6.1.

On the other hand, even the approach described above is limited in power. The following example suggests that in order to get a possibility result closer to the (γ,δ)-PassiveUNC simulation bound $\delta=2\gamma(1-\gamma)$ from [DKS99], one has to find different reduction methods and/or analytical tools.

Example 6.2 Let $\gamma_0=0.33$, $\delta_0=0.4$. For a SGWOT obtained from (γ_0,δ_0)-PassiveUNC the following holds: $p_s(\gamma_0,\delta_0)+q_r(\gamma_0,\delta_0)+2\epsilon(\delta_0)\approx 0.949$. It can be shown by brute force analysis that whatever sequence of reductions S-, R- and E-Reduce applied with whatever parameters, it always results at some point a SGWOT with $p_s+q_r+2\epsilon\geq 1$. In other words, given the original SGWOT, the reductions always generate a primitive such that the WOT corresponding to it (in the sense of Lemma 6.15) is trivial.

We stress that in contrast to a (p,q,ϵ)-WOT with $p+q+2\epsilon\geq 1$, a SGWOT with $p_s+q_r+2\epsilon\geq 1$ is not known to be trivial; however, it seems to be a very strong indication that OT cannot be based on such a SGWOT.

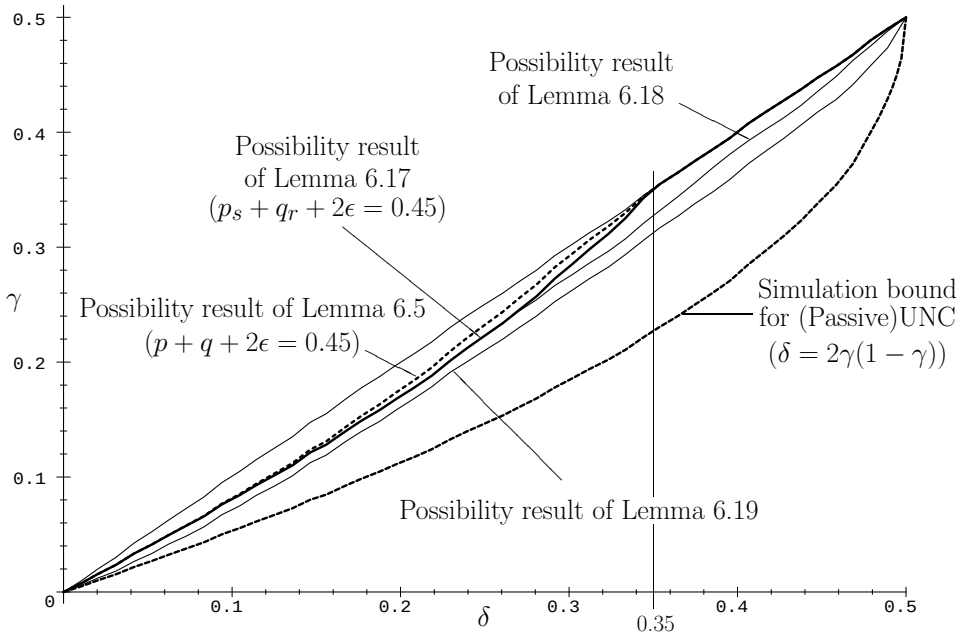

Figure 6.1: Positive results on OT from (γ, δ)-PassiveUNC

6.6 Concluding Remarks

6.6.1 More Applications

In this work, we have shown how to transform any OT protocol secure against passive adversaries given access to a PassiveUNC into one that is secure against active adversaries given access to a standard UNC. This is possible since any non-trivial UNC allows for bit commitment as it was shown in [DKS99]. Our transformation is general enough to be applicable to a wider class of two-party protocols. Applying it to a passively secure protocol π implementing task T given access to a PassiveUNC produces an actively secure protocol π' that implements T given access to a UNC, however, π' may fail with non-negligible probability. When T is OT, this can be "cleaned up" using the techniques described in this chapter, in general T can be any task where such "cleaning" is possible.

6.6.2 Open Question

We show that our approach for constructing OT based on UNC has limits that even a more tight analysis (compared to [DKS99]) cannot overcome. Thus, a "grey" range (which is between the possibility result of Lemma 6.19 and the simulation bound on Figure 6.1) is left where neither positive nor negative results are known to apply. Closing this gap is the open problem suggested by our work.

Bibliography

[Bea91] Beaver, D.: Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. In: J. of Cryptology, vol. 4, no. 2 (1991) 75–122

[BG89] Beaver, D., Goldwasser, S.: Multiparty Computation with Faulty Majority. In: Advances in Cryptology–CRYPTO '89. LNCS, vol. 435. Springer-Verlag (1990) 589–590

[BMM99] Beimel, A., Malkin, T., Micali, S.: The All-or-Nothing Nature of Two-Party Secure Computation. In: Advances in Cryptology–CRYPTO '99. LNCS, vol. 1666. Springer-Verlag (1999) 80–97

[BBCM95] Bennett, C.H., Brassard, G., Crépeau, C., and Maurer, U.M.: Generalized Privacy Amplification. IEEE Transactions on Information Theory, Vol. 41, Num. 6. IEEE Press (1995) 1915–1923

[BBCS92] Bennett, C.H., Brassard, G., Crépeau, C., and Skubiszewska, M.-H.: Practical quantum oblivious transfer. In: Advances in Cryptology–CRYPTO '91. LNCS, vol. 576. Springer-Verlag (1992) 351–366

[BBR86] Bennett, C.H., Brassard, G., and Robert, J.-M.: Privacy amplification by public discussion. SIAM J. on Computing, 17 (2). SIAM (1988) 210–229

[BGGHKMR88] Ben-Or, M., Goldreich, O., Goldwasser, S., Håstad, J., Kilian, J., Micali, S. and Rogaway, P. Everything Provable Is Provable in Zero-Knowledge. In: Advances in Cryptology–CRYPTO '88. LNCS, vol. 403. Springer-Verlag (1990) 37–56

[BM04] Ben-Or, M., Mayers., D.: General Security Definition and Composability for Quantum & Classical Protocols. e-Print archive, Quantum Physics. (Available from: http://arxiv.org/pdf/quant-ph/0409062)

[Bla87] Blahut, R.E.: Theory and practice of information theory. Addison-Wesley, Reading (1987)

[Blu81] Blum, M.: Three applications of the oblivious transfer: Part I: Coin flipping by telephone; part II: How to exchange secrets; Part III: How to send certified electronic mail. Technical report, Department of EECS, University of California, Berkeley (1981)

[Blu82] Blum, M.: Coin Flipping by Telephone. In: Proc. 24th IEEE Spring Computer Conference. IEEE Press (1982) 133–137

[BCC88] Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. of Computer and System Sciences, vol. 37, no. 2. Elsevier (1988) 156–189

[BC97] Brassard, G., Crépeau, C.: Oblivious transfers and privacy amplification. In: Advances in Cryptology–EUROCRYPT '97. LNCS, vol. 1233. Springer (1997) 334–347

[BCJL93] Brassard, G., Crépeau, C., Josza, R., and Langlois, D.: A quantum bit commitment scheme provably unbreakable by both parties. In: 34th IEEE FOCS. IEEE Press (1993) 362–371

[BCR86] Brassard, G., Crépeau, C., and Robert, J.-M.: All-or-Nothing Disclosure of Secrets. In: Advances in Cryptology–CRYPTO '86. LNCS, vol. 263. Springer-Verlag (1987) 234–238

[BCS96] Brassard, G., Crépeau, C., Sántha, M.: Oblivious Transfers and Intersecting Codes. IEEE Transactions on Information Theory, vol. 42, no. 6. (1996) 1769–1780

[BCW03] Brassard, G., Crépeau, C., and Wolf, S.: Oblivious transfers and privacy amplification. In: J. of Cryptology, vol. 16, no. 4 (2003) 219–237

[Cac97] Cachin, C.: Entropy measures and unconditional security in cryptography. Ph. D. Thesis. ETH Zürich, Hartung-Gorre Verlag, Konstanz (1997)

[Cac98] Cachin, C.: On the Foundations of Oblivious Transfer. In: Advances in Cryptology–EUROCRYPT'98. LNCS, vol. 1403. Springer-Verlag (1998) 361–374

[Can00] Canetti, R.: Security and Composition of Multi-party Cryptographic Protocols. J. of Cryptology, vol. 13, no. 1 (2000)

[Can01] Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd Symposium FOCS, IEEE (2001) 136–145

[Can01a] Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. Cryptology ePrint Archive: Report 2000/067. (Avaliable from: http://eprint.iacr.org/2000/067)

[Can03] Canetti, R.: Universally Composable Signatures, Certification and Authentication. Cryptology ePrint Archive: Report 2003/239. (Avaliable from: http://eprint.iacr.org/2003/239)

[Cré87] Crépeau, C.: Equivalence between two flavours of oblivious transfer. In: Advances in Cryptology–CRYPTO '87. LNCS, vol. 293. Springer-Verlag (1988) 350–354

[Cré89] Crépeau, C.: Verifiable Disclosure of Secrets and Applications. In: Advances in Cryptology–EUROCRYPT '89. LNCS, vol. 434. Springer-Verlag (1990) 150–154

[Cré90] Crépeau, C.: Correct and Private Reductions Among Oblivious Transfers. Ph. D. Thesis. MIT (1990)

[Cré97] Crépeau, C.: Efficient Cryptographic Primitives Based on Noisy Channels. In: Advances in Cryptology–EUROCRYPT '97. LNCS, vol. 1233. Springer-Verlag (1997) 306–317

[CCD88] Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Proc. 20th ACM STOC. ACM Press (1988) 11–19

[CCM98] Cachin, C., Crépeau, C., and Marcil, J.: Oblivious Transfer with a Memory-Bounded Receiver. In: Proc. 39th IEEE FOCS, IEEE (1998) 493–502

[CF85] Cohen, J.D., Fischer, M.J.: A Robust and Verifiable Cryptographically Secure Election Scheme. In: Proc. 26th IEEE FOCS. IEEE Press (1985) 372–382

[CFN88] Chaum, D., Fiat, A., and Naor, M.: Untraceable Electronic Cash. In: Advances in Cryptology–CRYPTO '88. LNCS, vol. 403. Springer-Verlag (1990) 319–327

[CDG87] Chaum, D., Damgård, I., and van de Graaf, J.: Multi-party Computations Ensuring Privacy of Each Party's Input and Correctness of the Result. In: Advances in Cryptology–CRYPTO '87. LNCS, vol. 293. Springer-Verlag (1988) 87–119

[CK88] Crépeau, C., Kilian, J.: Achieving Oblivious Transfer Using Weakened Security Assumptions. In: Proc. 29th IEEE FOCS. IEEE Press (1988) 42–52

[CMW04] Crépeau, C., Morozov, K., Wolf, S.: Efficient Unconditional Oblivious Transfer from Almost any Noisy Channel. Accepted to Fourth Conference on Security in Communication Networks '04, Amalfi (Italy), September, 2004

[CT91] Cover, T.M. and Thomas, J.A.: Elements of Information Theory. Wiley (1991)

[CW79] Carter, J.L., Wegman, M.N.: Universal Classes of hash functions. J. of Computer and System Sciences, vol. 18. Elsevier (1979) 143–154

[DFMS04] Damgård, I., Fehr, S., Morozov, K. and Salvail, L.: Unfair Noisy Channels and Oblivious Transfer. In: Theory of Cryptography Conference TCC '04. LNCS, vol. 2951. Springer-Verlag (2004) 355–373 (Full version is avaliable from: http://www.brics.dk/RS/03/36)

[DH76] Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory, vol. 22, no. 6. (1976) 644–654

[DKS99] Damgård, I., Kilian, J., Salvail, L.: On the (Im)possibility of Basing Bit Commitment and Oblivious Transfer on Weakened Security Assumptions. In: Advances in Cryptology–EUROCRYPT '99. LNCS, vol. 1592. Springer-Verlag (1999) 56–73

[EGL83] Even, S., Goldreich, O., and Lempel, A.: A randomized protocol for signing contracts. In: Proceedings CRYPTO '82. Plenum Press (1983) 205–210

[Fan61] Fano, R.M.: Transmission of Information. MIT Press (1961)

[For66] Forney, G.D.: Concatenated codes. MIT Press (1966)

[Fel68] Feller, W.: An introduction to probability theory. Wiley (1968)

[Gol01] Goldreich, O.: Foundations of Cryptology: Volume 1 – Basic Tools. Cambridge University Press (2001)

[GMR85] Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. In: Proc. 17th ACM STOC. ACM Press (1985) 291–304

[GMW86] Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing but the Validity of the Assertion, and the Methodology of Cryptographic Protocol Design. In: Proc. 27th IEEE FOCS. IEEE Press (1986) 174–187

[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to Play Any Mental Game, or: A completeness theorem for protocols with honest majority. In: Proc. 19th ACM STOC. ACM Press (1987) 218–229

[HILL91] Håstad, J., Impagliazzo, R., Levin, L.A., and Luby, M.: Construction of a pseudo-random generator form any one-way function. Technical report TR-91-068, International Computer Science Institute, Berkley, CA (1991)

[Kil88] Kilian, J.: Founding Cryptography on Oblivious Transfer. In: 20th ACM STOC. ACM Press (1988) 20–31

[Kil91] Kilian, J.: A General Completeness Theorem for Two-Party Games. In: 23rd ACM STOC. ACM Press (1991) 553–560

[Kil00] Kilian, J.: More General Completeness Theorems for Secure Two-Party Computation. In: 32nd ACM STOC. ACM Press (2000) 316–324

[KM00] Korjik. V., Morozov, K.: Non-asymptotic Bit Commitment Protocol Based on Noisy Channels. In: Proc. 20th Biennial Symposium on Communications. Queen's University, Kingston, Ontario, Canada (2000) 74–78

[KM01] Korjik, V., Morozov, K.: Generalized Oblivious Transfer Protocols Based on Noisy Channels. In: Proc. Workshop MMM ACNS 2001. LNCS, vol. 2052. Springer-Verlag (2001) 219-229

[Lin03] Lindell, Y.: Composition of Secure Multi-Party Protocols. LNCS, vol. 2815. Springer-Verlag, Heidelberg (2003)

[MR91] Micali, S. and Rogaway, P.: Secure Computation. In: Advances in Cryptology–CRYPTO '91. LNCS, vol. 576. Springer-Verlag (1991) 392–404

[MR95] Motwani, R., Raghavan, P.: Randomized algorithms. Cambridge University Press (1995)

[MS77] MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland (1977)

[Nao87] Naor, M.: Bit Commitment using Pseudo-Randomness. In: Advances in Cryptology–CRYPTO '89. LNCS, vol. 435. Springer-Verlag (1990) 128–136

[NP00] Naor, M., Pinkas, B.: Distributed Oblivious Transfer. In: Advances in Cryptology–Asiacrypt '00. LNCS, vol. 1976. Springer-Verlag (2000) 200–219

[PW72] Peterson, W.W., Weldon, E.J.: Error-correcting codes. MIT Press, Cambridge MA (1972)

[PW00] Pfitzmann, B., Waidner, M.: Composition and Integrity Preservation of Secure Reactive Systems. In: Proc. 7th ACM Conference on Computer and Communications Security. ACM Press (2000) 245–254

[Rab81] Rabin, M.O.: How to Exchange Secrets by Oblivious Transfer. Technical Memo TR-81, Aiken Computation Laboratory, Harvard University (1981)

[Rén61] Rényi, A.: On measures of entropy and information. In: Proc. 4th Berkley Symposium on Mathematical Statistics and Probability (Berkley), vol. 1. Univ. of Calif. Press (1961) 547–561

[Rén70] Rényi, A.: Probability theory. North-Holland, Amsterdam (1970)

[Riv99] Rivest, R.: Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer. Unpublished manuscript (1999). (Available from: http://theory.lcs.mit.edu/~rivest/Rivest-commitment.pdf)

[Rom90] Rompel, J.: Techniques for Computing with Low-Independence Randomness. Ph. D. Thesis. MIT (1990)

[Sha49] Shannon, C.E.: Communication theory of secrecy systems. Bell Systems Technical Journal, no. 28 (1949) 656–715

[SW02] Stebila, D., Wolf, S.: Efficient Oblivious Transfer From Any Non-Trivial Binary-Symmetric Channel. In: International Symposium on Information Theory (ISIT) (2002) 293

[Wie83] Wiesner, S.: Conjugate Coding. SIGACT News, no. 15(1). ACM Press (1983)

[WC81] Wegman, M.N., Carter, J.L.: New hash-functions and their use in authentication and set equality. J. of Computer and System Sciences, 22. Elsevier (1981) 265–279

[WNI03] Winter, A., Nascimento, A.C.A., Imai, H.: Commitment Capacity of Discrete Memoryless Channels. In: Cryptography and Coding. LNCS, vol. 2898. Springer-Verlag (2003) 35–51

[WN04] Winter, A., Nascimento, A.C.A.: Oblivious transfer from any genuine noise. Unpublished manuscript (2004)

[Yao82] Yao, A.C.: Protocols for Secure Computations. In: Proc. 23rd IEEE FOCS. IEEE Press (1982) 160–164

Index