Unfair Noisy Channels and Oblivious Transfer *

Ivan Damgård[1], Serge Fehr[2]**, Kirill Morozov[1], and Louis Salvail[1]***

[1] BRICS†, FICS ‡, Aarhus University, Denmark
{ivan,kirill,salvail}@brics.dk
[2] ACAC§, Department of Computing, Macquarie University, Australia
sfehr@ics.mq.edu.au

Abstract. In a paper from EuroCrypt'99, Damgård, Kilian and Salvail show various positive and negative results on constructing Bit Commitment (BC) and Oblivious Transfer (OT) from Unfair Noisy Channels (UNC), i.e., binary symmetric channels where the error rate is only known to be in a certain interval $[\gamma..\delta]$ and can be chosen adversarily. They also introduce a related primitive called $PassiveUNC$. We prove in this paper that any OT protocol that can be constructed based on a $PassiveUNC$ and is secure against a passive adversary can be transformed using a generic "compiler" into an OT protocol based on a UNC which is secure against an active adversary. Apart from making positive results easier to prove in general, this also allows correcting a problem in the EuroCrypt'99 paper: There, a positive result was claimed on constructing from UNC an OT that is secure against active cheating. We point out that the proof sketch given for this was incomplete, and we show that a correct proof of a much stronger result follows from our general compilation result and a new technique for transforming between weaker versions of OT with different parameters.

1 Introduction

Bit Commitment (BC) and Oblivious Transfer (OT) are the most fundamental primitives in cryptographic protocol design [8, 1, 3, 9, 10]. But in a scenario with only two players, neither primitive can be implemented with unconditional security based only on standard, error free communication. Even quantum communication does not help [14, 13]. However, Crépeau and Kilian have shown that both primitives can be implemented based on a binary symmetric channel (BSC) [5]. A BSC is a channel for transmitting single bits, and for every bit transmitted, the channel decides with some fixed probability to flip the bit before it is

* This is the full version of [6].
** Most of this work was done while at BRICS, Aarhus University, Denmark.
*** Research funded by European project PROSECCO.
† Basic Research in Computer Science (www.brics.dk), funded by the Danish National Research Foundation.
‡ FICS, Foundations in Cryptography and Security, funded by the Danish Natural Sciences Research Council.
§ Centre for Advanced Computing - Algorithms and Cryptography.

given to the receiver. Unfortunately, results based on BSCs do not give realistic security guarantees. The reason for this is that one must expect that a cheating player will try to influence the channel and have this work to his/her advantage, for instance by lowering the noise rate in order to learn more than expected about what the other party sent or received. Note that one can always hide the fact that the channel was made less noisy by pretending to have sent(received) a more noisy signal than the one actually sent(received). Moreover, even in the absence of such attacks, it is hardly realistic to assume that the noise rate is known exactly.

In [7], Damgård, Kilian and Salvail introduce the Unfair Noisy Channel (UNC) as a model of a noisy channel that is more realistic in cryptographic applications than a BSC. A (γ, δ)-UNC is basically a BSC, where, however, the noise rate is only known to be in a certain interval $[\gamma..\delta]$, and where if the sender or receiver has been corrupted by an adversary, the adversary can set the noise rate to any desired value in the interval. So a UNC models active cheating directed against the way a physical channel works in order to manipulate the error rate. If the channel is a radio link, for instance, the adversary could invest in more sophisticated receiving equipment without telling the other party and thereby lowering the noise rate from his point of view. However, it may still be realistic to assume that he cannot remove *all* noise from the channel, so such a case can be captured in the UNC model.

Another primitive was also introduced, namely a (γ, δ)-PassiveUNC. This is a BSC with error rate δ, but where the adversary gets for every transmission some side information z with the property that given z, the bit received/sent by the other (honest) player can be guessed with error probability γ. In other words, knowledge of z brings the error rate down to γ from the adversary's point of view. This models a passive, i.e., "honest but curious" adversary, who measures somewhere "in the middle" of the channel, and then later uses the information obtained to compute data he should not have access to.

In [7], it was proved that Bit Commitment (BC) can be implemented with unconditional security based on a (γ, δ)-UNC if and only if the interval $[\gamma..\delta]$ is not too wide, more precisely, if and only if $\delta < 2\gamma(1-\gamma)$. It was also shown that one cannot base Oblivious Transfer (OT) on a (γ, δ)-UNC (nor on a PassiveUNC) if $\delta \geq 2\gamma(1-\gamma)$. On the positive side, it was shown that if γ and δ satisfy a rather complex condition (stronger than $\delta < 2\gamma(1-\gamma)$), then OT (with *passive* security) can be based on a (γ, δ)-PassiveUNC.

Finally, it was claimed that this same result also holds when using a (γ, δ)-UNC, and with security against *active* cheating. This was based on a standard idea where the players use bit commitments to commit to all private data, including what is sent and received on the channel, and then use generic zero-knowledge techniques to demonstrate that they follow the protocol. This technique indeed works assuming that we can force a cheating player to commit to the bits he actually sends or receives over the channel (except with arbitrarily small probability). This assumption is true for a BSC: for instance the sender S can be instructed to commit to bits $b_i, i = 1..n$, and send them over the BSC with

noise rate, say, δ. Having received bits $\hat{b}_i, i = 1..n$, the receiver R then asks to have all committed bits opened except one, say b_j. If S was honest, we expect that a fraction of about δ of the opened bits will be different from the received bits $\hat{b}_i, i = 1..n$. So R is instructed to reject if the fraction of disagreement is significantly larger than δ. If R does not reject, this means intuitively that he believes that the committed bit b_j really is the bit that was sent over the channel and resulted in R receiving $\hat{b}_j$. This is justified since it follows from standard probability theory that the probability of having b_j different from the j'th bit actually sent and still have the receiver accept, can be made arbitrarily small by increasing n.

Unfortunately, no such technique can work for a UNC. We show below that for any protocol that aims to implement a "committed UNC", the probability of error is at least a constant, namely $(\delta - \gamma)/(1 - 2\gamma)$. This problem was not taken care of in [7].

In this paper, we show a different (and correct) way to apply the idea of using commitments and zero-knowledge proofs to enforce correct behavior. This turns out to lead to a result that is much more general than what was claimed in [7] and which can be informally stated as follows: Any two-party protocol that, based on a (γ, δ)-PassiveUNC, implements an OT secure against passive cheating, can be transformed using a generic "compiler" into a protocol that uses a (γ, δ)-UNC for communication and builds an OT secure against active cheating.

The opposite direction of this result is also true, and trivial to prove. So this implies that, to prove positive or negative results, on building OT from UNC or PassiveUNC, we can now concentrate only on the case of PassiveUNC and passive cheating - which is clearly much simpler. It also immediately implies a complete proof of the claim made in [7].

In the final part of the paper we exploit this, and a new technique for transforming between the weaker versions of OT, in order to prove a stronger positive OT result than the one claimed in [7]. In other words, there is now a much larger range of (γ, δ)-values for which we can implement OT based on a (γ, δ)-UNC. For instance we can now show that robust OT follows from a (γ, δ)-UNC with *any* value of δ between 0 and 1/2, provided γ is close enough to δ.

2 Models of Communication and Adversaries

Our protocols throughout the paper 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 message delivery for all channels such that failure to send a message can be detected.

In order to specify formally the channels and reductions we study, we will use the universally composable framework of Canetti [2]. 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 some 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 2-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". We express this in the UC framework by assuming an infinitely powerful non-adaptive adversary who from the start has corrupted no one, or either A or B. While we believe that our results extend to adaptive adversaries, we do not prove or claim this in this paper. Furthermore, if the noisy channel is a UNC, then the adversary is assumed to be active, i.e., can decide the corrupted player's behavior. If the channel is a PassiveUNC, the adversary is passive.

Another consequence of being in the two-player case, is that we do not think of our protocols as subroutines in a multiplayer 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. At the cost of more complex proofs, our results extend to the case where the adversary always eavesdrops the error free channel.

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 formalized 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 tell if it is in the real-life or the ideal model. For details, see [2].

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 behavior of the honest players be as in the real attack, then Z will not be able to tell any difference.

The noisy channels we study in this paper can very conveniently be modeled as ideal functionalities, and reductions that build one type of channel from another can be proved secure in this framework. Since the results we prove are information theoretic in nature, we modify the UC model as given in [2] by allowing our adversaries and simulators infinite computing power - but we stress that honest players can execute our protocols efficiently.

3 Some functionalities

We can now specify our basic types of channels precisely but for completeness we start by describing the functionality for standard (1-out-of-2) OT as well as for a weak version as introduced in [7] with parameters $0 \leq p, q \leq 1$ and $0 \leq \epsilon \leq 1/2$:

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, 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.

For later convenience, we call the receiver's choice c the *selection bit* and the bit b_{1-c} (which is not revealed to the receiver) the *secret bit*.

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 $Pr(\tilde{b}_c \neq b_c) = \epsilon$ and sends it to the receiver. Additionally, if the sender is corrupted, then with probability p it sends c to the sender, and if the receiver is corrupted, then with probability q it sends b_{1-c} to the receiver.

A (γ, δ)-UNC is specified by the following functionality.

Functionality (γ, δ)**-UNC**

Send b**:** The issuer of the Send command is called the sender, the other party is the receiver. On receipt of this command, 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 if $\gamma \leq \epsilon \leq \delta$. If not, the command is ignored. Otherwise, it chooses a random bit b', such that $Pr(b' = 1) = \epsilon$, 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.

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

We introduce some notation that will be convenient: if we cascade a BSC with error rate x and a BSC with error rate y, the result is again a BSC, we define $x \boxplus y$ to be the resulting error rate, $x(1-y)+(1-x)y$. Note that the operator $\boxplus$ is commutative, associative and satisfies that if $|x-x'| < \nu$, then $|x \boxplus y - x' \boxplus y| < \nu$ for all y.

Functionality (γ,δ)-PassiveUNC

Send b: The issuer of the Send command is called the sender, the other party is the receiver. On receipt of this command, the functionality chooses random bits b', b'', such that $Pr(b'=1)=\gamma$ and $Pr(b''=1)=\nu$, where $\nu \boxplus \gamma = \delta$. This ensures that $Pr(b' \oplus b''=1)=\delta$. 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. Intuitively, given z, the noise rate goes down to γ.

We need to consider the use of commitments and zero-knowledge proofs in our protocols. This can also be modeled by an ideal functionality, where one commits simply by giving the bit to the trusted party, who will then later open it on request from the committer. Furthermore, the trusted party will confirm that committed bits satisfy a given formula, if this is indeed true.

Functionality Commit-and-prove (CaP)

Commit cID, b: Receiving this command, where cID is a bitstring 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 all players.

Open cID: if cID, b has been received earlier from the player issuing this command, send b to all 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 all players. Else, send $(Fail, L, \Phi)$.

As bit commitment scheme in our protocols, we will use the UNC-based construction from [7], which works assuming $\delta < 2\gamma(1-\gamma)$ which we will assume throughout. This scheme is statistically close to perfect, regardless of A and B's computing power. 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 [11]). It follows that in any protocol where we assume access to a UNC, we may assume also a CaP without loss of generality.

A final functionality that will come in handy is the ability to choose random bits and numbers with a prescribed distribution:

Functionality RandomChoice

Flip sID, ν**:** Here sID is a session ID and ν must be a probability. Once the functionality has received this command from every player containing identical values of sID, ν, it chooses a bit b at random such that $Pr(b = 1) = \nu$ and sends b to all 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 every player containing identical values of sID, j, it chooses i uniformly from $[0..j-1]$ and sends i to all 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 realized 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 any time.

4 Committed (Passive)UNC

We first define informally the notion of a *committed UNC*. This is a protocol for players A, B, using a (γ, δ)-UNC and an error free channel. We will 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, 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 resp. 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 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 [7] 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, i.e., are honest, but curious. 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 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 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 as essentially like a PassiveUNC, *even in presence of active cheating.* We model this by an ideal functionality called $(\gamma, \delta, q())$-Committed Passive-UNC (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 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 $Pr(b' = 1) = \gamma$ and $Pr(b'' = 1) = \nu$, where $\nu \boxplus \gamma = \kappa'$. This ensures that $Pr(b' \oplus b'' = 1) = \kappa'$. 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 all players, and ignores all further commands until the receiver sends a "ReciptID" command.

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

Commit cID, b: Receiving this command, where cID is a bitstring 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 all players.

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

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.

Send (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 $Pr(c = 1) = \mu$.
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 $Pr(n_j = 1) = \gamma$. By the choice of c, and random choice of j, we have

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

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.

In the appendix we show the theorem below. First, some terminology to state the result: we say 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 2. *The Committed Passive UNC protocol securely realizes the $(\gamma, \delta, q())$-CPUNC functionality when given access to ideal (γ, δ)-UNC, CaP and Random-Choice 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 1. The last claim in the theorem is a way to state in the UC framework the traditional completeness property for a 2-party protocol: if both players are honest, the protocol completes successfully with overwhelming probability.

5 From passive to active security

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

Theorem 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.*

So 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 section 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 trick: 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, you use 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 weak form of OT (which then implies standard OT), namely a (p, q, ϵ)-WOT as defined in Section 3.

Lemma 1. *$\bar{\pi}$ 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 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 $adv_\pi(k) = \sum_v Pr(V = v) \cdot adv_\pi(k, v)$ 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 E be the event that V takes a bad value. Then clearly, 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 $i = 0..t(k)$, 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 (γ, δ)-CPUNC for the rest. Then hybrid 0 is $\bar{\pi}$ while hybrid $t(k)$ is π. When executing hybrid i, we define 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 E_i occurs. Of course $\epsilon_{t(k)} = Pr(E) \leq \sqrt{adv_\pi(k)}$. 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: In [7], a reduction is shown that implements OT 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 the above lemma, 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 Extended positive results

In this section, we shall assume the result of Theorem 3 and focus on reducing OT to (γ, δ)-PassiveUNC securely against passive adversaries. The strategy of [7] is as follows. First, the (γ, δ)-PassiveUNC is used to construct an imperfect version of OT which may leak information about the parties' private inputs. This imperfect OT is modeled by a WOT. OT is then shown to be reducible to WOT for certain values of (γ, δ).

However, WOT does not precisely capture the imperfect OT obtained in the construction: 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. As a consequence, in order to fit the imperfect OT into the WOT model, it is assumed in [7] that every time the dishonest sender/receiver gets some information about the selection/secret bit, he actually gets full information. Hence, the information leakage is overestimated in [7]. We introduce a new *Generalized Weak Oblivious Transfer (GWOT)* primitive which allows to model imperfect OTs which leak information about the parties' private inputs in a much more general way than WOTs, without overestimating the information leakage. In particular, it precisely captures the imperfect OT resulting from the construction of [7]. 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 functionality of the following kind.

Functionality $\left(\{(s_i, \alpha_i)\}_{i=1}^N; \{(r_i, \beta_i)\}_{i=1}^N; \epsilon\right)$-GWOT

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 $Pr(\tilde{b}_c \neq b_c) = \epsilon$ and sends it to the receiver. Additionally, if the sender is corrupted, then it chooses $I \in \{1, \ldots, N\}$ and $\tilde{c} \in \{0, 1\}$ such that $Pr(I = i) = s_i$ and $Pr(\tilde{c} \neq c \,|\, I = i) = \alpha_i$, and it sends I and $\tilde{c}$ to the sender. And/or, if the receiver is corrupted, then it chooses $I \in \{1, \ldots, N\}$, and $\tilde{b}_{1-c} \in \{0, 1\}$ such that $Pr(I = i) = r_i$ and $Pr(\tilde{b}_{1-c} \neq b_{1-c} \,|\, I = i) = \beta_i$, and it sends I and $\tilde{b}_{1-c}$ to the receiver.

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 functionality'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 $\left(\{(p, 0), (1-p, 1/2)\}; \{(q, 0), (1-q, 1/2)\}; \epsilon\right)$-GWOT.

It will be convenient to introduce a GWOT of a very particular form, a *Special Generalized 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 $0 \leq \alpha, \beta \leq 1/2$,

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

Consider the reduction of WOT to (γ,δ)-PassiveUNC given in Appendix A of [7]. As mentioned above, this construction actually results in a GWOT (which is modeled by a WOT by giving away information to the adversary). As a matter of fact, as 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). For convenience, we write $\mu = \frac{\delta-\gamma}{1-2\gamma}$, such that $\gamma \boxplus \mu = \delta$. The proof of the Lemma follows by straightforward analysis of reduction WOTfromPassiveUNC of [7].

Lemma 2. *When run with a (γ,δ)-UNC, reduction WOTfromPassiveUNC defined in [7] produces a $\big((s,\alpha),(r,\beta),\epsilon\big)$-SGWOT with the following parameters:*

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

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

$$\epsilon = \frac{\delta^2}{\delta^2+(1-\delta)^2}. \quad (3)$$

We have expressed the parameters of $\big((s,\alpha),(r,\beta),\epsilon\big)$-SGWOT with that of the underlying (γ,δ)-PassiveUNC. Now we would like to exploit the machinery of [7] in order to reduce OT to SGWOT. A composition of three basic reductions is used in order to transform a WOT into an OT. The first reduction, S-Red(l) decreases the sender's information about the selection bit by executing WOT l times such that the final selection bit is the parity of all selection bits used during the l executions (this reduction was introduced in [5]). The second reduction, R-Red(l), decreases the receiver's information about the bit that was not selected by encoding it into the parity of l transmissions. The final reduction, E-Red(l), decreases the error rate by executing l identical transmissions through a WOT. Every of these reductions transforms the WOT into a new one (with new parameters), and it is shown in [7] that for certain initial parameters the sequence of WOTs converges to an OT (in some well defined meaningful sense).

In [7], the $\big((s,\alpha),(r,\beta),\epsilon\big)$-SGWOT obtained after invoking WOTfromPassiveUNC was modeled 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 GWOTs, resulting 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 [7] is executed, the shape of the GWOTs becomes quickly very complex and difficult to analyze.

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 [7] 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 BSCs, 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 3) to capture the relative difficulty to generate OT using the reduction considered in [7]. Intuitively, we say that $S \preceq S'$ if S can be transformed into S' by removing BSCs in S and replacing each of them by a Bernoulli distribution over 2 BSCs such that the *average* guessing probability for the bit sent through S is the same as when sent through S'.

Definition 1. *Let $S = \{(p_i, \epsilon_i)\}_{i=1}^N$ and S' be two probability distributions over BSCs. 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 BSCs 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 thorough S or through S^*.

The next lemma, proved in Appendix B, 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 3. *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 $R \preceq R'$.*

One application of Lemma 3 allows to improve the analysis of [7]. As we have seen in Lemma 2, the imperfect OT obtained from a UNC using reduction WOTfromPassiveUNC produces a $\big((s, \alpha), (r, \beta), \epsilon\big)$-SGWOT. Using Lemma 3 it is straightforward 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 5 of [7] (OT is possible based on (p, q, ϵ)-WOT if $p + q + 2\epsilon \leq 0.45$) to the transformed SGWOT results in the following Lemma.

Lemma 4. *The reduction from OT to WOT of [7] implements OT from any $\big((s, \alpha), (r, \beta), \epsilon\big)$-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 4 and 2, gives directly the following result:

Lemma 5. *OT may 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 (1)–(3).*

Note that [7] only guarantees that OT can be achieved if $p + q + 2\epsilon \leq 0.45$ where $p = 1 - s$ and $q = 1 - r$. Hence, the possibility range given in Lemma 5 strictly contains the one obtained in [7].

Despite this improvement, Lemma 5 still shares the following restriction with [7]. OT cannot be provably achieved for $\delta > 0.35$ even when γ is almost equal to δ (i.e. the resulting UNC has almost no unfairness) since in that case $\epsilon > 0.45$ (see Figure 1). This stands somewhat in contrast to the fact that OT can be achieved based on any (non-trivial) BSC [4, 12, 15]. Hence, one would 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 is indeed true.

Lemma 6. *There exists a reduction from 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 (1)–(3).*

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 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 1). This further improves on [7].

Proof. We implement a $\big((s,\alpha),(r,\beta),\epsilon\big)$-SGWOT from the (γ,δ)-PassiveUNC according to Lemma 2. Then, by Lemma 3, we convert it into a (p_s,q_r,ϵ)-WOT before applying the reduction E-Red(l) [7] with parameter l. As shown in [7], this results in a $(1-(1-p_s)^l, 1-(1-q_r)^l, \frac{\epsilon^l}{\epsilon^l+(1-\epsilon)^l})$-WOT. The claim now follows from the above. □

It can be shown by straightforward calculations that the new possibility range includes UNC's for which the techniques of [7] results in a "simulatable" WOT (i.e., a trivial WOT), that is, could not be used to implement OT (see Lemma 1 from [7]). In other words, our approach allows to implement and prove secure OT in a range where it is *provably impossible* using the techniques of [7]. The following example illustrates this.

Example 1. Let $\gamma_0 = 0.39$, $\delta_0 = 0.4$ be the parameters of a PassiveUNC. The $\big(p(\gamma_0,\delta_0), q(\gamma_0,\delta_0), \epsilon(\delta_0)\big)$-WOT obtained from a (γ_0,δ_0)-PassiveUNC the crude way (by giving away all partial information to the adversary as in [7]) 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 $\big(p(\gamma_0,\delta_0), q(\gamma_0,\delta_0), \epsilon(\delta_0)\big)$-WOT using S-, R- and E-Red. At the same time, the $\big(p_s(\gamma_0,\delta_0), q_r(\gamma_0,\delta_0), \epsilon(\delta_0)\big)$-WOT (obtained according Lemma 3) 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 5 implies OT.

There exists an even larger range than the one described in Lemma 6 for which a possibility result can be shown. This follows from the fact that the approach of Lemma 6 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 preserving the SGWOT through the sequence of reductions.

The problem is that the reductions do not preserve the SGWOT per se but produce more complex GWOTs with a quickly growing set of parameters. An approach is to use Lemma 3 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 $\big((s, \alpha), (r, \beta), \epsilon\big)$-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 7. *There exists a range of values (γ, δ) which do not satisfy the conditions of Lemma 6 but where OT can still be implemented from such (γ, δ)-UNC'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 are not satisfied with respect to this SGWOT. Nonetheless, the sequence of reductions "EERSRESERRSESRERSESERRS" (each with parameter $l = 2$) produces as output a SGWOT with $p_s + q_r + 2\epsilon = 0.329$ which implies OT according Lemma 5. □

Using brute-force analysis, it is possible to find experimentally the range for which the reduction considered in Lemma 7 produces OT. The new range is depicted on Figure 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 [7], one has to find different reduction methods and/or analytical tools.

Example 2. Let $\gamma_0 = 0.33$, $\delta_0 = 0.4$. A SGWOT based on (γ_0, δ_0)-PassiveUNC has the potential $p_s(\gamma_0, \delta_0) + q_r(\gamma_0, \delta_0) + 2\epsilon(\delta_0) \approx 0.949$. 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 $p_s + q_r + 2\epsilon \geq 1$.

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 proven to be simulateable; however, it seems to be a very strong indication that OT cannot be based on such a SGWOT.

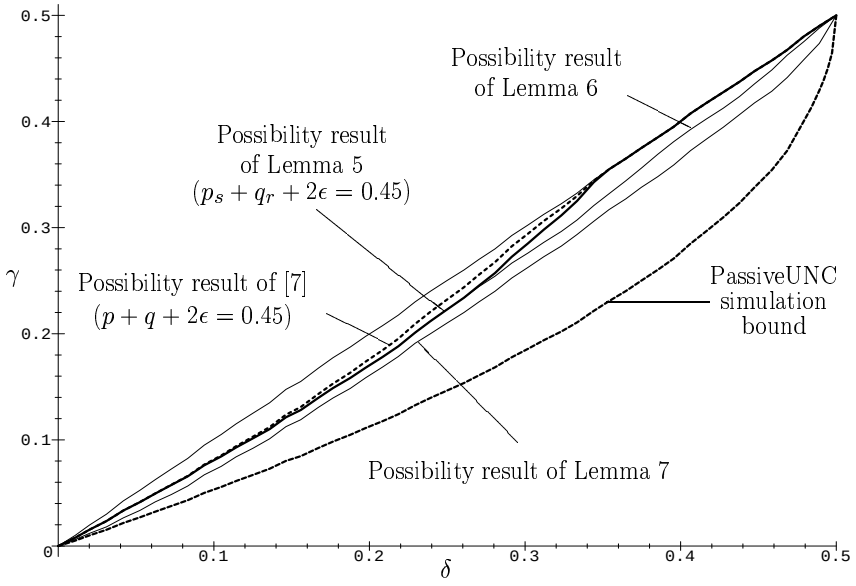

Fig. 1. Positive results on OT from (γ, δ)-PassiveUNC

7 Conclusion and Open Questions

In this paper, 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 [7]. Our transformation is general enough to be applicable to a wider class of 2-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 (1/poly) probability. When T is OT, this can be cleaned up using known techniques, in general T can be any task where such "cleaning" is possible.

We have also provided a more refined analysis for the reduction of OT to (γ, δ)-UNC introduced in [7]. As a result, OT is now possible based on a significantly larger range of (γ, δ) than what was known before. Unfortunately, we also show the approach has limits that even a more careful analysis cannot overcome. Thus, a grey area is left where no positive or negative results are known to apply. Closing this gap is the obvious open problem suggested by this work.

References

1. Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. of Computer and System Sciences, 37(2). Elsevier (1988) 156–189

2. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd Symposium FOCS, IEEE (2001) 136–145
3. Chaum, D., Damgård, I., 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 (1987) 462
4. Crépeau, C.: Efficient Cryptographic Protocols Based on Noisy Channels. In: Advances in Cryptology–EUROCRYPT'97. LNCS, vol. 1233. Springer-Verlag (1997) 306–317
5. Crépeau, C., Kilian, J.: Achieving Oblivious Transfer Using Weakened Security Assumptions. In: 29th Symposium FOCS, IEEE (1988) 42–52
6. Damgård, I., Fehr, S., Morozov, K. and Salvail, L.: *Unfair Noisy Channels and Oblivious Transfer*. In: Theory of Cryptography Conference TCC '04. LNCS, vol. ???. Springer-Verlag (2004) ???–???
7. 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
8. Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing but the Validity of the Assertion, and the Methodology of Cryptographic Protocol Design. In: 27th Symposium FOCS. IEEE (1986) 174–187
9. Goldreich, O., Micali, S., Wigderson, A.: How to Play Any Mental Game. In: 19th ACM STOC. ACM Press (1987) 218-229
10. Kilian, J.: Founding Cryptography on Oblivious Transfer. In: 20th ACM STOC. ACM Press (1988) 20–31
11. Kilian, J.: A Note on Efficient Proofs and Arguments. In: 24th ACM STOC. ACM Press (1992) 723–732
12. 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
13. Lo, H.–K., Chau, H. F.: Is Quantum Bit Commitment Really Possible?. Physical Review Letters, vol. 78, no 17, (1997) 3410–3413
14. Mayers, D.: Unconditionally Secure Quantum Bit Commitment is Impossible. Physical Review Letters, vol. 78, no 17, (1997) 3414–3417
15. Stebila, D., Wolf, S.: Efficient Oblivious Transfer From Any Non-Trivial Binary-Symmetric Channel. In: International Symposium on Information Theory (ISIT) (2002) 293

A Proof of Theorem 2

We begin with some technical lemmas. First a basic fact from probability theory which follows from the Hoefding inequality:

Lemma 8. *Assume we do l independent experiments, such that in the i'th experiment the event E occurs with probability q_i, and let $q = \sum_i q_i/l$. Let β be the fraction of the l experiments in which E was observed. Then* $Pr(|\beta - q| > \nu) \leq 2\exp(-l\nu^2/2)$

Note that when $q_1 = q_2 = ... = q_l = q$, this reduces to (a variant of) the well known Bernstein's law of large numbers or the Chernoff bound.

Now, 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: *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 $\kappa \boxplus \mu = \delta + 1/p(k)$.

As usual, a probability is called negligible in k if as a function of k, it converges to 0 faster than any polynomial fraction. Now Lemma 8 trivially implies that κ is a good estimate of $err(\bar{C}, \hat{C})$:

Lemma 9. *For any given $\bar{C}, \hat{C}$, $Pr(|\kappa - err(\bar{C}, \hat{C})| > 1/p(k)$ is negligible in k.*

Lemma 8 also implies:

Lemma 10. *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 $Pr(|err(\bar{C}, \hat{C}) - \epsilon \boxplus \gamma| > 1/p(k))$ is negligible in k.*

Proof. We apply Lemma 8 with $l = N$, where the i'th experiment consists of flipping the i'th bit of $\tilde{C}$ with probability to get the i'th bit of $\hat{C}$. The event E is 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 11. *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 12. *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 13. *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 14. *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 positions shows disagreement is δ and this happens independently for each of the $kp(k)^2$ sampled positions, so we have $Pr(abort) \leq Pr(|\kappa - \delta| > 1/p(k)) \leq 2\exp(-k/2)$.

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 15. *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 8 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 straightforward 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 now finally ready to show that we have a good implementation of the CPUNC functionality. Some terminology to state the result: we say 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 4. *The Committed Passive UNC protocol securely realizes the CPUNC functionality when given access to ideal UNC, CaP and RandomChoice functionalities, and if we choose $p(k) = 4q(k)$. Moreover, for the case where both players are honest, the simulator is non-blocking.*

Remark 2. The last claim in the theorem is a way to state in the UC framework the traditional completeness property for a 2-party protocol: if both players are honest, the protocol completes successfully with overwhelming probability.

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 corrupt 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 $b_j = \tilde{b}_j$ – 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' 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 receiver 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 13 we have $|\delta - \epsilon' \boxplus \gamma \boxplus \mu| < 1/q(k)$, except with negligible probability, 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 $Pr(z \neq b) = \epsilon' \boxplus \mu$. 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 $Pr(\tilde{b}_j \oplus b' \oplus c \neq \hat{b}) = \gamma$. 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, $Pr(z \neq \hat{b}) = \gamma$.

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 15, this only happens with negligible probability. Furthermore, the simulator only blocks the CPUNC if α is illegal, or if the value of κ is too large. By lemmas 15,14, 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.

B Proof of Lemma 3

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), or 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 1. 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 1.

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, independently for $j = 1, \ldots, l$, a channel (index) $I_j \in \{1, \ldots, N\}$ is chosen such that $Pr(I_j = i) = p_i$ 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 $Pr(I'_j = +) = \delta$ 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 straightforward 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 analyze. 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 probability coincide by the assumption on δ, ϵ, ϵ^+ and ϵ^- posed in Definition 1, 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

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

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

$$Pr(c=0) = \frac{\rho_0(1-\epsilon)+\rho_1\epsilon}{\rho_0+\rho_1} = \frac{(1-\epsilon)+\alpha\epsilon}{1+\alpha} \quad \text{and}$$
$$Pr(c=1) = \frac{\rho_0\epsilon+\rho_1(1-\epsilon)}{\rho_0+\rho_1} = \frac{\epsilon+\alpha(1-\epsilon)}{1+\alpha}\,.$$

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 $0 \le \alpha \le 1$, and hence

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

Similarly, it holds that

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

Therefore,

$$\begin{aligned} guess' &= \frac{1}{1+\alpha}\cdot\left(\alpha\epsilon^+ + \delta\cdot\max\{\epsilon^+,\alpha(1-\epsilon^+)\} + (1-\delta)\cdot\max\{\epsilon^-,\alpha(1-\epsilon^-)\}\right) \\ &= \frac{1}{1+\alpha}\cdot\left(\alpha\epsilon^+ + \max\{\delta\epsilon^+, \delta\alpha(1-\epsilon^+)\} + \max\{(1-\delta)\epsilon^-, (1-\delta)\alpha(1-\epsilon^-)\}\right) \\ &\ge \frac{1}{1+\alpha}\cdot\left(\alpha\epsilon^+ + \max\{\delta\epsilon^+ + (1-\delta)\epsilon^-, \delta\alpha(1-\epsilon^+) + (1-\delta)\alpha(1-\epsilon^-)\}\right) \\ &= \frac{1}{1+\alpha}\cdot\left(\alpha\epsilon + \max\{\epsilon, \alpha(1-\epsilon)\}\right) \\ &= guess \end{aligned}$$

This had to be shown. □