Oblivious Transfer from Any Non-Trivial Elastic Noisy Channels via Secret Key Agreement

Ignacio Cascudo, Ivan Damgård, Felipe Lacerda, Samuel Ranellucci*
{ignacio,ivan,lacerda,samuel}@cs.au.dk
Aarhus University

Abstract

A (γ, δ)-elastic channel is a binary symmetric channel between a sender and a receiver where the error rate of an honest receiver is δ while the error rate of a dishonest receiver lies within the interval $[\gamma, \delta]$. In this paper, we show that from *any* non-trivial elastic channel (i.e., $0 < \gamma < \delta < \frac{1}{2}$) we can implement oblivious transfer with information theoretic security. This was previously (Khurana et al., Eurocrypt 2016) only known for a subset of these parameters. Our technique relies on a new way to exploit protocols for information-theoretic key agreement from noisy channels. We also show that information theoretically secure commitments where the receiver commits follow from any non-trivial elastic channel.

Keywords: oblivious transfer, elastic channels, key agreement, commitments

1 Introduction

In this paper we consider oblivious transfer (OT), a well known two-party cryptographic primitive. In oblivious transfer, a sender has two messages and a receiver chooses to learn one of them. The receiver gains no information about the other message, while the sender does not know which of the messages the receiver has learned. Oblivious transfer is an important primitive because it is sufficient for information-theoretic secure computation [Kil88].

However, information-theoretic secure computation and therefore oblivious transfer are well known to be impossible if sender and receiver communicate in the plain model, with no other additional resource available. Therefore, several alternative models have been studied where information-theoretically secure oblivious transfer is possible because we assume additional resources.

One such assumption is the existence of a noisy channel between the sender and the receiver. It was shown in [CK88] that binary symmetric channels are in fact enough to realize oblivious transfer. A binary symmetric channel is one where each bit sent is flipped with a certain probability, known as the error rate of the channel. More efficient constructions, and different variants of noisy channels, were provided in subsequent papers, such as [BCS96, Cré97, DKS99, DFMS04, CMW05, CS06, PDMN11, IKO+11].

In particular, it was realised that it is problematic to assume that we are given a noisy channel with known and fixed parameters, such that the OT protocol we construct is allowed to depend on the parameter values. One reason for this is that it can be very hard to reliably

[*] The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation and from the Center for Research in Foundations of Electronic Markets (CFEM), supported by the Danish Strategic Research Council. Samuel Ranellucci is supported by European Research Council Starting Grant 279447.

estimate the parameters of a real channel. Another, more serious problem is that by fixing the parameters we are implicitly assuming that the adversary cannot change them. This is clearly unrealistic, and was the main motivation for introducing *unfair noisy channels* (UNC) in [DKS99]. In this model, the channel is a binary symmetric channel where, however, an adversary who corrupts one of the two parties can also choose the error rate to be within some range $[\gamma, \delta]$. For $\delta \geq 2\gamma(1-\gamma)$, the channel is easily seen to be trivial (it can be simulated from noiseless communication). It was shown in [DKS99] that information theoretically oblivious transfer follows from UNC for a certain subset of the possible non-trivial parameter choices, while information theoretically secure commitments follow from any non-trivial UNC.

Elastic channels (EC), a relaxation of unfair noisy channels, have been introduced in [KMS15]. For an EC, the noise can only be reduced by an adversary who corrupts the receiver. More precisely, given $0 \leq \gamma < \delta \leq 1/2$, a (γ, δ)-elastic channel is one where the communication between the sender and an honest receiver has error rate δ, but a dishonest receiver may reduce this to be in the interval $[\gamma, \delta]$. Clearly, in this setting, $\delta = 1/2$ would correspond to a channel where all information is lost for the honest receiver, while $\gamma = 0$ would yield a channel where a dishonest receiver has full information about the messages sent by the sender. Clearly we cannot implement oblivious transfer in either case, and hence these channels are deemed trivial.

It was shown in [KMS15] that commitments where the sender commits follow from any non-trivial EC, and that oblivious transfer follow from EC for a certain subset of parameters, which is larger than in the case of an UNC. More specifically they show that $\delta \leq \ell(\gamma)$ where $\ell(\gamma) := \left(1 + (4\gamma(1-\gamma))^{-1/2}\right)^{-1}$ is sufficient.

It is of course interesting that going from UNC to EC allows a larger range of parameters from which we can get OT. However, for both channels, we are still left with a "grey area" of parameter values where we do not know if OT is possible. One might say that we still do not know if an EC is *fundamentally and qualitatively different* from a UNC as far as OT is concerned. Moreover, for commitments, we know that we can have the sender commit, but since an EC is asymmetric w.r.t what corrupted senders and receivers can do, it is not clear that we can get commitments where the receiver commits for any non-trivial EC.

Our contribution. In this paper, we make progress on the above questions. First, we close the gap left open in [KMS15] and show that information theoretically secure oblivious transfer follows from any non-trivial EC. Along the way, we also construct commitments where the receiver commits, from any non-trivial EC.

Our main technical contribution is a new way to exploit existing constructions of key agreement protocols towards implementing OT. More specifically, we consider a key agreement protocol between two parties (Alice and Bob) in the following model: Alice can send messages to Bob through a binary symmetric channel C with error rate δ, and the adversary Eve will receive what Alice sends via an independent binary symmetric channel with error rate $\gamma' \in [\gamma, \delta]$. On top of this, Alice and Bob may also communicate via a public error-free channel. Several key agreement protocols exist in this model [Mau93]. The main idea is to use the public channel to identify transmissions where Alice and Bob are more likely to agree on what was sent on the noisy channel. Because Eve's channel is independent, this may create a situation where Eve has a disadvantage compared to Bob, even if her noise rate is initially smaller.

In this work, we consider key agreement protocols that are secure in the usual sense: Alice and Bob agree on the output, and Eve gets essentially no information on the key. But in addition, we require an extra property we call *emulatability*: We can replace Bob by a "fake" Bob', who gets *no information* on what Alice sends on the noisy channel (but Eve gets information with error rate γ' as usual). Still, Bob' can complete the conversation on the public channel such that neither Alice nor Eve can distinguish Bob' from Bob. As we explain later, we can modify the key agreement protocol from [Mau93] so that it is emulatable. We show that an oblivious transfer protocol secure against semi-honest adversaries can be constructed from *any* emulatable key agreement protocol. Furthermore, by using information-theoretical commitments where the committing party is the receiver (which can be constructed from any non-trivial EC, as we will show) we can upgrade our protocol to achieve security against a malicious receiver too. Finally, we show how to achieve security against a malicious sender in the case where our emulatable key agreement protocol is the modification of the protocol from [Mau93].

Technical overview. To give an intuition of how our protocol works, consider first the case of semi-honest security where a semi-honest receiver reduces the error rate to the minimal value γ (which is without loss of generality).

We turn a emulatable key agreement (KA) protocol as described above into an OT protocol as follows. The sender and the receiver engage in two independent instances (indexed respectively by 0 and 1) of the key agreement protocol above. In both cases, the sender from the OT protocol takes the role of Alice in the KA protocol, while the receiver does the following: in the instance of the KA protocol corresponding to his selection bit b, he acts as Bob would, while in the other instance, he acts as Bob' (so in particular his actions are independent of what he receives from the sender on the EC). Finally Alice sends her messages m_0, m_1 one-time padded respectively with k_0 and k_1, each of these keys obtained in the corresponding key agreement protocol.

Now, an honest receiver will learn m_b as he should, which follows from correctness of the KA protocol. Second, a corrupt sender cannot learn the choice bit b. This follows from the emulatability property of the KA protocol: the sender cannot distinguish in which of the two instances she is interacting with the real Bob. Finally, a corrupt receiver cannot learn m_{1-b}. This follows from the fact that, in the instance of the KA corresponding to $1 - b$, the view of the receiver is the same as the view of Eve, namely he sees everything Alice sends with error rate γ, and he sees the public discussion (the fact that he generates that discussion himself by running Bob' makes no difference). One can then show that emulatability implies that this view is distributed identically to the case where Eve watches Alice interact with Bob, and the usual definition of key agreement security guarantees that this is independent from the exchanged key k_{1-b}.

Security in the malicious case is more involved. First, we need to ensure that the malicious receiver follows the protocol. It turns out to be sufficient that the receiver proves that for one of the KA instances, the messages he sends on the public channel are generated by Bob', of course without revealing which one. To this end we can use the fact that commitments where the committing party is the receiver also follow from any EC (see below) and, via a known reduction, zero-knowledge proofs on committed values follow as well. Thus, we are doing something very similar to the GMW compiler. As a result we get a protocol that is secure against a semi-honest sender and a malicious receiver.

To further protect against a malicious sender, we execute many instances of the OT. The receiver checks the statistics of what he receives on the EC and discards instances that are too far from what he expects to see from an honest sender. This creates a protocol where the sender will (at least sometimes) have non-trivial uncertainty about the choice bit. We can now use standard techniques to clean this up to get a secure OT.

As for our construction on receiver commitments from any non-trivial EC, we observe that the commitment protocol from [DKS99] (that was designed for a UNC) can be modified to work for an EC. All we essentially have to do is to choose the parameters correctly. On the one hand, handling an EC is harder because δ and γ are much further apart than for a UNC, however, on the other hand an EC is easier because one party has to live with the large noise rate even if he is corrupt. Intuitively, the observation is that these two issues balance each out so that (almost) the same protocol still works.

Outline. In Section 2 we define the basic functionalities we will deal with for the remainder of the paper, namely oblivious transfer and the elastic channel. In Section 3, we introduce the notion of emulatable key agreement, as well as a protocol that implements it. Emulatable key agreement is used in Section 4 to implement an OT protocol that is secure against semi-honest adversaries. this protocol is then used in Section 5 in the construction of a protocol secure against a malicious receiver. Finally, in Section 6 we present a construction that builds upon the one of Section 5 to obtain security against malicious adversaries.

2 Preliminaries

2.1 Security model

We prove our protocols secure in the Universal Composability framework introduced in [Can01]. This model is explained in Appendix A.

2.2 Oblivious transfer

Oblivious transfer is a two-party primitive where one party (the sender) inputs two messages and the other party (the receiver) chooses to receive one—and only one—of them. Crucially, the sender does not learn the receiver's choice, and the receiver does not learn the message it did not choose. This primitive is formalized in the primitive below. Note that the description includes an adversary $\mathcal{A}$, which can corrupt parties.

Functionality $\mathcal{F}_{\texttt{OT}}$ (Oblivious transfer)

$\mathcal{F}_{\texttt{OT}}$ runs with two parties: a sender and a receiver.

Send: Upon receiving $(\texttt{send}, \text{sid}, m_0, m_1)$ from the sender: store (sid, m_0, m_1) and send $(\texttt{sent}, \text{sid})$ to $\mathcal{A}$.

Receipt: Upon receiving $(\texttt{choice}, \text{sid}, b)$ from the receiver: if a message of the form (sid, m_0, m_1) has been stored, send $(\texttt{receipt}, m_b)$ to the receiver.

2.3 Elastic channel

A (γ, δ)-elastic channel, as introduced in [KMS15], is a binary symmetric channel with crossover probability δ where a receiver that has been corrupted by the adversary can choose to reduce the crossover probability to a level ν with $\gamma \leq \nu \leq \delta$. In the functionality below, we define a more general version where the channel is composed by ℓ binary symmetric channels (all with crossover probability ν).

Functionality $\mathcal{F}_{\texttt{EC}}(\gamma, \delta)$ (Elastic channel)

$\mathcal{F}_{\texttt{EC}}$ runs with parties P_1, P_2 and eavesdropper $\mathcal{A}$ as follows:

Initialization: $\nu \leftarrow \delta$

Noise: Upon receiving $(\texttt{noise}, \bar{\nu})$ from $\mathcal{A}$, if the receiver is corrupt and $\gamma \leq \bar{\nu} \leq \delta$ then set $\nu \leftarrow \bar{\nu}$.

Send: On $(\texttt{send}, \mathrm{sid}, m)$ from the sender, where $m \in \{0,1\}^\ell$, produce $\bar{m}$ by flipping each bit of m independently with probability ν. Then send the message $(\texttt{sent}, \mathrm{sid})$ to $\mathcal{A}$ and the message $(\texttt{sent}, \mathrm{sid}, \bar{m})$ to the receiver.

3 Emulatable key agreement

Key agreement is the problem where two parties, Alice and Bob, want to establish a common key (a random element from $\{0,1\}^\ell$) so that an eavesdropper Eve has no information about this key. In other words, the goal is to implement the following functionality $\mathcal{F}_{\texttt{KA}}$.[1]

Functionality $\mathcal{F}_{\texttt{KA}}$ (Key agreement)

$\mathcal{F}_{\texttt{KA}}$ runs with security parameter u, parties P_1, P_2 and eavesdropper $\mathcal{A}$ as follows:

Establish: Upon receiving $(\texttt{establish}, \mathrm{sid}, P_i, P_j)$ from P_i, store (sid, P_i, P_j) and send (sid, P_i, P_j) to $\mathcal{A}$. If there is already a stored (sid, P_j, P_i), choose $k \leftarrow_R \{0,1\}^t$, send $(\texttt{sent}, 1^t)$ to $\mathcal{A}$ and send $(\texttt{key}, \mathrm{sid}, k)$ to P_1, P_2.

In this section, we consider the scenario in which Alice can communicate to Bob via a wiretap channel $\mathcal{F}_{\texttt{C}}$ where each bit is flipped (independently) with probability δ. Eve can obtain another noisy version of this communication, where each bit is flipped with probability γ and this noise is independent from Bob's. Furthermore, there is a feedback public channel $\mathcal{F}_{\texttt{Pub}}$ through which Alice and Bob can communicate.

Functionality $\mathcal{F}_{\texttt{C}}$ (Wiretap channel)

$\mathcal{F}_{\texttt{C}}$ runs with parameters $\gamma, \delta \in (0, 1/2)$, message size ℓ, parties P_1, P_2 and eavesdropper $\mathcal{A}$ as follows:

Send: Upon receiving $(\texttt{send}, \mathrm{sid}, P_1, P_2, m)$ where $m \in \{0,1\}^\ell$, for each $k \in \{1, \ldots, \ell\}$:

1. Produce $\bar{m}$ by flipping each bit of m independently with probability δ. Furthermore, produce $\widetilde{m}$ by flipping each bit of m independently with probability γ.
2. Send $(\texttt{sent}, \mathrm{sid})$ to P_1, $(\texttt{receipt}, \mathrm{sid}, \bar{m})$ to P_2 and $(\texttt{receipt}, \mathrm{sid}, \widetilde{m})$ to $\mathcal{A}$.

Functionality $\mathcal{F}_{\texttt{Pub}}$ (Public channel)

$\mathcal{F}_{\texttt{Pub}}$ runs with message size ℓ, parties P_1, P_2 and eavesdropper $\mathcal{A}$ as follows:

Send: Upon receiving $(\texttt{send}, \mathrm{sid}, P_i, P_j, m)$ where $m \in \{0,1\}^\ell$, send $(\texttt{sent}, \mathrm{sid})$ to P_i and $(\texttt{receipt}, \mathrm{sid}, m)$ to P_j and $\mathcal{A}$.

[1] In the remainder of this section, we interchangeably call the parties Alice, Bob, Eve or respectively $P_1, P_2, \mathcal{A}$.

In this setting, we are interested in key agreement protocols with an additional property that we call *emulatability.* A key agreement protocol π is emulatable if, in addition to implementing the key agreement functionality as it should, the role of Bob can be simulated by some entity $\mathcal{E}$, the emulator, that learns *no information* about the messages transmitted through $\mathcal{F}_{\texttt{C}}$, other than their lengths, and neither Alice nor Eve can distinguish whether Alice is interacting with Bob or with $\mathcal{E}$.

We formalize this below. We first define a functionality $\mathcal{F}_{\texttt{DC}}$ that models a dummy channel whose task is to erase every information sent through the channel $\mathcal{F}_{\texttt{C}}$ except for the length of the messages.

Functionality $\mathcal{F}_{\texttt{DC}}$ (Dummy channel)

$\mathcal{F}_{\texttt{DC}}$ runs with message size ℓ and parties P_1, P_2 as follows:

Send: Upon receiving $(\texttt{send}, \mathrm{sid}, m)$ from P_1 where $m \in \{0,1\}^\ell$:

If no such command has already been sent, send $(\texttt{sent}, \mathrm{sid}, |1|^\ell)$ to P_2.

Otherwise, ignore the command.

Definition 3.1 *A key agreement protocol π between Alice and Bob using a wiretap channel $\mathcal{F}_{\texttt{C}}$ and a public channel $\mathcal{F}_{\texttt{Pub}}$ is* emulatable *if:*

1. *It realizes the functionality $\mathcal{F}_{\texttt{KA}}$. That is, there exists a simulator $\mathcal{S}$ such that for all eavesdroppers $\mathcal{A}$,*
$$\pi \diamond \mathcal{F}_{\texttt{C}} \diamond \mathcal{F}_{\texttt{Pub}} \equiv_{\mathcal{A}} \mathcal{F}_{\texttt{KA}} \diamond \mathcal{S}.$$

2. *There exists an emulator $\mathcal{E}$ such that the following happens: suppose that we consider the protocol π' where we replace Bob by $\mathcal{F}_{\texttt{DC}} \diamond \mathcal{E}$, i.e., $\mathcal{E}$ is linked to $\mathcal{F}_{\texttt{C}}$ via the dummy channel $\mathcal{F}_{\texttt{DC}}$, and Alice acts as in protocol π, while in both cases the eavesdropper $\mathcal{A}$ receives information from $\mathcal{F}_{\texttt{C}}$ and $\mathcal{F}_{\texttt{Pub}}$. Then from the point of view of Alice and all eavesdroppers $\mathcal{A}$, the protocol executions of π and π' are indistinguishable.*

   That is, we have
$$\pi \diamond \mathcal{F}_{\texttt{C}} \diamond \mathcal{F}_{\texttt{Pub}} \equiv_{\mathrm{Alice},\mathcal{A}} \pi' \diamond C \diamond \mathcal{F}_{\texttt{Pub}}.$$

We will need the following property later on.

Proposition 3.2 *Suppose that a key agreement protocol π is emulatable. Then for any eavesdropper $\mathcal{A}$, if Alice is executing the protocol π' with the emulator $\mathcal{E}$ as in the definition, $\mathcal{A}$ obtains no information about Alice's output.*

This is because, if $\mathcal{A}$ could obtain any information about Alice's output in the execution of π', then either she would be able to obtain information about Alice's output in the execution of π (contradicting property 1 of emulatability) or she would be able to distinguish π and π' (contradicting property 2).

3.1 The emulatable key agreement protocol

We now describe an emulatable key agreement protocol for a wiretap channel $\mathcal{F}_{\texttt{C}}$ with $\gamma < \delta$, that is, for which the channel to the eavesdropper Eve is more reliable than the channel to Bob.

This is a small modification of a key agreement protocol from [Mau93].

For each γ, δ, the protocol specifies large enough numbers $s, \ell, n \in \mathbb{N}$, to be determined below. In addition, $\ell = 2m + 1$ is an odd number. The protocol consists of three phases: advantage distillation, information reconciliation and privacy amplification [BBR88, Mau93, BS94, BBCM95, CM97, MW00, RW05, HR11].

The goal of the advantage distillation step is to create a conceptual channel between Alice and Bob which is more reliable than the one between Alice and Eve. In our protocol, this step proceeds as follows. Alice samples n random bits b_i and encodes each bit b as a bitstring v in $\{0,1\}^\ell$ by selecting uniformly at random a set $\mathcal{J} \subseteq \{1, \dots, \ell\}$ of size m and setting the j-th coordinate of v to be $1 - b$ if $j \in \mathcal{J}$ and b if $j \notin \mathcal{J}$. Note that this means $m + 1$ of the bits of the encoding equal b while the other m bits equal $1 - b$. Now, if for a given sent bit b, Bob receives a message of the form $(c, c, \dots, c)$ for some c, we say Bob accepts the bit b and c is his guess about b. Now Alice creates the bitstring $b_{i_1} b_{i_2} \dots b_{i_s}$ given by the first s bits accepted by Bob and Bob creates the bitstring $c_{i_1} c_{i_2} \dots c_{i_s}$ of his guesses. They both disregard the remaining bits. The intuition here is that, even though Eve has more information over messages sent over the wiretap channel than Bob has, she has less information over the messages accepted by Bob.

The information reconciliation step is carried out over the public channel. After this step, Alice and Bob will share a common bitstring with overwhelming probability, and Eve is still guaranteed to have some uncertainty about it. Alice and Bob now apply privacy amplification to obtain a random string about which Eve has *no* information.

Both the information reconciliation and privacy amplification steps can be done by having only Alice send information over the public channel. This is important to guarantee the emulatability property. Furthermore, both steps can be done together as one step.

This description is formalized below. (For simplicity, we omit the description of the "establish" step introduced in the functionality of Section 3.)

Protocol $\pi_{\mathtt{KA}}$ (Emulatable key agreement)

Parameters:

- σ: security parameter.
- $\ell := \ell(\gamma, \delta)$: an odd natural number which only depends on γ and δ.
- $m := \frac{\ell - 1}{2}$.
- $s \in \omega(\sigma)$.
- $t \in \Theta(\sigma)$.
- $n > \lceil s / (\delta(1 - \delta)^m) \rceil$.
- $0 < \epsilon < \frac{1}{2} - \delta$: a small constant.

Let $\mathfrak{h}$ denote the binary entropy function. Let $\mathcal{H}_1 : \{0,1\}^s \rightarrow \{0,1\}^{s \cdot \mathfrak{h}(\delta + \epsilon) + \sigma}$, $\mathcal{H}_2 : \{0,1\}^s \rightarrow \{0,1\}^t$ families of 2-universal hash functions.

Advantage distillation:

Alice:

Select $b_1, \dots, b_n \in_R \{0,1\}$.

For $i \in \{1, \dots, n\}$:

- Select a set $\mathcal{J}_i \subseteq \{1, \dots, \ell\}$ of size m uniformly at random.
- Set v_i to be the bitstring in $\{0,1\}^\ell$ such that $v_i[j] = 1 - b_i$ for $j \in \mathcal{J}_i$ and $v_i[j] = b_i$ for $j \notin \mathcal{J}_i$ where $v_i[j]$ denotes the j-th coordinate of v_i.
- Send $(\mathtt{send}, \text{Alice}, \text{Bob}, \text{sid}_i, v_i)$ to $\mathcal{F}_{\mathtt{C}}$.

Bob:

For $i \in \{1, \dots, n\}$, await $(\mathtt{receipt}, \text{Alice}, \text{Bob}, \text{sid}_i, \bar{v}_i)$ from $\mathcal{F}_{\mathtt{C}}$.

Construct the set $\mathcal{I} \subseteq \{1, \dots, n\}$ consisting of the indices i for which $\bar{v}_i = c_i^\ell$ for some $c_i \in \{0,1\}$

Encode the set $\mathcal{I}$ as a bit string u and send $(\mathtt{send}, \text{sid}, \text{Bob}, \text{Alice}, u)$ to $\mathcal{F}_{\mathtt{Pub}}$.

Alice:

Await $(\mathtt{sent}, \text{sid}, \text{Bob}, \text{Alice}, u)$ from $\mathcal{F}_{\mathtt{Pub}}$.

Alice $\leftrightarrow$ Bob:

Alice sets $X^s = (b_{i_1}, b_{i_2}, \dots, b_{i_s})$ and Bob sets $Y^s = (c_{i_1}, c_{i_2}, \dots, c_{i_s})$, where $i_1, \dots, i_s$ are the first s indices in $\mathcal{I}$.

Information reconciliation and privacy amplification:

Alice:

Sample $h_1 \in_R \mathcal{H}_1$, $h_2 \in_R \mathcal{H}_2$, send $(\mathtt{send}, \text{sid}, \text{Alice}, \text{Bob}, h_1, h_1(X^s), h_2)$ to $\mathcal{F}_{\mathtt{Pub}}$.

Output $h_2(X^s)$.

Bob:

Await $(\mathtt{send}, \text{sid}, \text{Alice}, \text{Bob}, h_1, h_1(X^s), h_2)$ from $\mathcal{F}_{\mathtt{Pub}}$.

Find the closest (in the Hamming metric) bitstring $\widetilde{X}^s$ to Y^s satisfying $h_1(\widetilde{X}^s) = h_1(X^s)$.

Output $h_2(\widetilde{X}^s)$.

We now highlight the differences we introduce with respect to the protocol in [Mau93] and the rationale behind them. To start with, [Mau93] considers a slightly different situation, in which Alice, Bob and Eve have all noisy versions of a common string at the beginning of the protocol, where each bit is independently flipped with probabilities α, δ and γ for Alice, Bob and Eve respectively.

In our scenario, we could reproduce the situation above by having Alice select a random bitstring r, introduce artificial noise to r (by flipping each bit of r with probability α), and send it through the wiretap channel. This would be a secure key agreement protocol. However, the protocol would not be emulatable: the reason is that the probability that Bob accepts a given instance depends on the number of bits that have been flipped by Alice. This information is known by Alice but cannot be determined precisely by the emulator. Hence, regardless of how we define the emulator, Alice will be able to distinguish when she is interacting with it or with Bob.

We solve this issue by having Alice flip a fixed number m of bits in each encoding of a bit.

Now Bob's response can be perfectly emulated, since from Alice's view, the probability that Bob accepts a given instance is always the same regardless of which subset of coordinates she has flipped.

If Alice introduced no bitflips at all, there would exist an emulator for the protocol; however, it can be seen that, in that case, the advantage distillation step would only give an advantage to Bob for a certain range of parameters; namely for $\delta < \left(1 + (4\gamma(1-\gamma))^{-1/2}\right)^{-1}$ (note this is exactly the same range of parameters for which an OT protocol for a (γ, δ)-elastic noisy channel is constructed in [KMS15]).

On the other hand, our protocol guarantees that for any values of γ and δ, we can take ℓ large enough so that after the advantage distillation step, the uncertainty of Bob about Alice's string is smaller than that of Eve.

In order to prove that our protocol is indeed an emulatable key agreement protocol, we introduce the following notation. Let X denote a variable with the uniform distribution over $\{0, 1\}$. Let Y and Z be the random variables that describe respectively the output bit c of Bob and the received bitstring of Eve (which is an element in $\{0, 1\}^\ell$) when Alice samples a bit b according to X, encodes it as in our protocol, sends it through the wiretap channel *and Bob accepts*. An important point to make is that, since the noise of Bob and Eve are independent, the probability distribution of Z would be the same if we removed the conditioning on Bob accepting the bit.

Let $S_0 \subseteq \{0, 1\}^\ell$ be the set of all bitstrings with the majority of the bits equal to 0, i.e., the set of all bitstrings with Hamming weight at most m and let S_1 be the set of all bitstrings with the majority of the bits equal to 1.

We will need the following lemma which intuitively means that, as ℓ grows, the probability that Eve receives a bitstring with the majority of its bits equal to 0 approaches 1/2 if Alice encoded a 0 (naturally an analogous statement holds if Alice encoded a 1).

Lemma 3.3 $\Pr[Z \in S_0 | X = 0] \to 1/2$ *as* $\ell \to \infty$.

Proof. Appendix B. △

Theorem 3.4 *The protocol* π_{KA} *is an emulatable key agreement protocol.*

Proof.

We first argue that, if we set paramaters adequately, the protocol is correct and secure, i.e., with overwhelming probability Alice and Bob have a common string at the end of the protocol about which Eve has a negligible amount of information.

Remember that for each index i, Alice encodes b_i as a bitstring containing $m+1$ bits equal to b_i and m bits equal to $(1 - b_i)$ and Bob accepts if he receives c_i^ℓ. Hence, the probability that Bob accepts i, i.e., the probability that an index i is in $\mathcal{I}$ is $p_{\text{accept}} = \delta^m (1-\delta)^{m+1} + \delta^{m+1}(1-\delta)^m = (\delta(1-\delta))^m$.

On the other hand, conditioned on Bob accepting index i, the probability that $c_i \neq b_i$ is

$$\frac{\delta^{m+1}(1-\delta)^m}{(\delta(1-\delta))^m} = \delta.$$

Furthermore these probabilities are independent from each i, so the advantage distillation step creates another conceptual noisy channel where Alice communicates s bits to Bob and the noise parameter is still δ (independently of how large ℓ is).

Hence if we set n slightly larger than $\lceil s/p_{\text{accept}} \rceil$ for large enough parameters, Bob will very high probability accept at least s bits, of which roughly $\delta \cdot s$ will be incorrect. By the results on information reconciliation in [BS94], if h_1 is chosen from the 2-universal family of hash functions $\mathcal{H}_1$, then Bob can correct to the right string X^s with very high probability given his original string, h_1 and $h_1(X^s)$. Hence both Alice and Bob will compute the same value $h_2(X^s)$ with high probability and hence the protocol is correct.

As for privacy, remember that X denotes the uniform distribution over $\{0,1\}$ and Z the variable that represents Eve's output when Alice chooses b according to X, encodes it, and sends it through the channel. Then

$$H_\infty(X|Z) = \sum_{z\in\{0,1\}^\ell} \Pr[Z=z]\cdot(-\log(\max_{b\in\{0,1\}} \Pr[X=b|Z=z])).$$

Now the maximum of $\Pr(X=b|Z=z)$ is reached for $b=0$ if $z\in S_0$ and for $b=1$ if $z\in S_1$. On the other hand, for every $z\in S_0$, we have $z' := (1,\dots,1)-z\in S_1$ and clearly, $\Pr[X=0|Z=z] = \Pr[X=1|Z=z']$. Hence we can write

$$H_\infty(X|Z) = \sum_{z\in S_0} 2\cdot\Pr[Z=z]\cdot(-\log\Pr[X=0|Z=z]).$$

Now, clearly $\sum_{z\in S_0} 2\cdot Pr[Z=z]=1$ and $-\log$ is a convex function. This means we can apply Jensen's inequality to get

$$H_\infty(X|Z) \geq -\log\left(\sum_{z\in S_0} 2\cdot\Pr[Z=z]\Pr[X=0|Z=z]\right).$$

Now we use that $\Pr[Z=z]\Pr[X=0|Z=z] = \Pr[X=0]\Pr[Z=z|X=0] = \frac{1}{2}\Pr[Z=z|X=0]$, so after summing over $z\in S_0$ we obtain:

$$H_\infty(X|Z)\geq -\log\Pr[Z\in S_0|X=0]\to 1$$

as $\ell\to\infty$ because of lemma 3.3. Since $\delta+\epsilon<1/2$, for large enough ℓ, we have $\mathfrak{h}(\delta+\epsilon)<H_\infty(X|Z)$ (remember $\mathfrak{h}(\cdot)$ denotes the binary entropy function).

Now let X^s, Y^s denote the random variables denoting the s bits outputted by Alice and Bob respectively and let Z^s be the variable representing the s bitstrings outputted by Eve. Then clearly $H_\infty(X^s|Z^s) = sH_\infty(X|Z) > s\mathfrak{h}(\delta+\epsilon)+t+3\sigma$ since $t,\sigma=o(s)$ and therefore $H_\infty(X^s|Z^s,h_1,h_1(X^s)) \geq H_\infty(X^s|Z^s) - s\mathfrak{h}(\delta+\epsilon)-\sigma > t+2\sigma$.

Now, the leftover hash lemma guarantees that conditioned on everything seen by Eve during the protocol, the distribution of $h_2(X^s)$ is $2^{-\sigma}$-close to the uniform distribution over $\{0,1\}^t$.

To show that the protocol is emulatable, we have to construct an emulator $\mathcal{E}$ that satisfies Property 2 in Definition 3.1. We note that the only information Bob sends to Eve is the description of the set $\mathcal{I}$ of indices for which Bob accepted Alice's message. We can construct an emulator for Bob thus. After $\mathcal{E}$ receives a message from the dummy channel $\mathcal{F}_{\texttt{DC}}$, it samples a random index set $\mathcal{I}\subseteq\{0,1\}^n$, where each index is chosen according to a Bernoulli distribution with parameter $p_{\texttt{accept}}$—the index is included in $\mathcal{I}$ if the trial succeeds. $\mathcal{E}$ then sends a description of $\mathcal{I}$ to Alice via $\mathcal{F}_{\texttt{Pub}}$. It is clear that such an emulator satisfies Property 2.

△

4 Semi-honest protocol

Now we present an OT protocol over the elastic channel $\mathcal{F}_{\mathtt{EC}}(\gamma, \delta)$ for semi-honest adversaries. We show that such an oblivious transfer protocol can be constructed from any emulatable key agreement protocol that works in the setting of Section 3 (where Alice, Bob and Eve are connected by a wiretap channel $\mathcal{F}_{\mathtt{C}}$ with the noise parameters being δ for Bob and γ for Eve).

The idea of the protocol is for sender and receiver to engage in two separate subprotocols. In one, they run the emulatable key agreement protocol with the sender acting as Alice and the receiver acting as Bob. In the other subprotocol, the sender follows again the key agreement protocol as Alice, whereas the receiver runs the emulator, according to Definition 3.1. The choice bit c determines whether the receiver will follow the protocol or act as the emulator. Here, the elastic channel is used as a conceptual wiretap channel $\mathcal{F}_{\mathtt{C}}$, where an honest receiver gets the output of the legitimate (noisier) channel, whereas an adversarial receiver gets the output of the less noisy channel.

To see why the protocol is secure, we note that since the key agreement protocol is emulatable, the sender does not know whether she is interacting with Bob (that is, whether she is engaging in the actual key agreement protocol) or with the emulator. Hence, she does not learn any information about the choice bit c. This guarantees the receiver's privacy.

On the other hand, by definition the emulator can generate the transcript for the key agreement protocol without knowing anything about the exchanged key. Therefore in this case the receiver has no information about the key output by Alice at the end of the key agreement protocol.

This proof sketch is formalized in Theorem 4.1, below.

Protocol $\pi_{\mathtt{OTSH}}$ (Semi-honest oblivious transfer)

Let $\pi_{\mathtt{KA}}$ be an emulatable key agreement protocol, as stated in Definition 3.1. We denote the sender's input as m_0, m_1 and denote the receiver's input as c.

Sender $\leftrightarrow$ Receiver:

Sender and receiver execute two copies π_0, π_1 of $\pi_{\mathtt{KA}}$, where the sender behaves in both as Alice. In π_c, the receiver acts as Bob and in π_{1-c}, it acts as the emulator $\mathcal{E}$ prescribed by $\pi'_{\mathtt{KA}}$.

Receiver:

On completion of π_0, π_1, record the output of π_c as k.

Sender:

Await k_0, k_1 from π_0, π_1.

Set $\bar{m}_i := m_i \oplus k_i$ for $i = 0, 1$.

Send $(\mathtt{send}, \mathrm{sid}_0, \bar{m}_0)$ and $(\mathtt{send}, \mathrm{sid}_1, \bar{m}_1)$ to $\mathcal{F}_{\mathtt{Pub}}$.

Receiver:

Await $(\mathtt{sent}, \mathrm{sid}_0, \bar{m}_0)$, $(\mathtt{sent}, \mathrm{sid}_1, \bar{m}_1)$ from $\mathcal{F}_{\mathtt{Pub}}$.

Output $m_c := \bar{m}_c \oplus k$.

Theorem 4.1 *The protocol $\pi_{\texttt{OTSH}}$ realizes $\mathcal{F}_{\texttt{OT}}$. That is, there exists a simulator $\mathcal{S}$ such that*

$$\pi_{\texttt{OTSH}} \diamond \mathcal{F}_{\texttt{EC}} \diamond \mathcal{F}_{\texttt{Pub}} \equiv_{\mathcal{Z}} \mathcal{F}_{\texttt{OT}} \diamond \mathcal{S}$$

for all semi-honest environments $\mathcal{Z}$.

Proof. For each activation, the environment $\mathcal{Z}$ chooses m_0, m_1, c. When interacting with the protocol, $\mathcal{Z}$ receives m'_c, and when interacting with $\mathcal{F}_{\texttt{OT}}$, it receives m_c. We note first that since π_c is an instance of $\pi_{\texttt{KA}}$, which implements $\mathcal{F}_{\texttt{KA}}$, we have $m'_c = m_c$. All that remains to be shown is that there exists a simulator for $\mathcal{F}_{\texttt{OT}}$ that can reproduce the view of the environment.

First, assume P_1 is corrupted, so that $\mathcal{Z}$ gets access to P_1's internal state. During the real execution, it gets access to k_0, k_1 (through P_1), $\bar{m}_0, \bar{m}_1$ plus the leakage from π_0 and π_1 (through the adversary $\mathcal{A}$, which interacts with $\mathcal{F}_{\texttt{EC}}$ and $\mathcal{F}_{\texttt{Pub}}$). At the end of the execution, it gets P_2's output, which is given by m_c.

In the ideal process, the simulator $\mathcal{S}$ corrupts P_1, so that it gets access to m_0, m_1. $\mathcal{S}$ proceeds as follows. First, it executes two copies of $\mathcal{F}_{\texttt{KA}} \diamond \mathcal{S}'$, where $\mathcal{S}'$ is the simulator for the key agreement protocol. By assumption, this internal simulator replicates the leakage from π_0 and π_1, which is relayed to $\mathcal{Z}$. Additionally, at the end of $\mathcal{F}_{\texttt{KA}}$'s execution, $\mathcal{S}$ gets two random keys, which we denote by k''_0, k''_1. It then computes $\bar{m}_i = m_i \oplus k''_i$ for $i = 1, 2$ and sends both to $\mathcal{Z}$. Finally, it sends m_0, m_1 to $\mathcal{F}_{\texttt{OT}}$, which will then send m_c to P_2. It is easy to see that $\mathcal{F}_{\texttt{OT}} \diamond \mathcal{S}$ provides $\mathcal{Z}$ with the same view as in the real protocol.

Now assume P_2 is corrupted. Throughout the real execution, $\mathcal{Z}$ gets access to k_c (through P_2), $\bar{m}_0, \bar{m}_1, m_c$ plus the leakage from π_0 and π_1 (through the eavesdropper $\mathcal{A}$). In the ideal process, $\mathcal{S}$ gets c by corrupting P_2. It proceeds as follows. It runs one copy of $\mathcal{F}_{\texttt{KA}} \diamond \mathcal{S}'$, obtaining a random key k''_c, and relays c to $\mathcal{F}_{\texttt{OT}}$. After P_2 receives m_c from $\mathcal{F}_{\texttt{OT}}$, $\mathcal{S}$ computes $\bar{m}_c = m_c \oplus k''_c$ and sends it to $\mathcal{Z}$. Clearly, m_c and $\bar{m}_c$ have the same distribution as in the real execution.

Finally, we look at the leakage from the execution of π_{1-c} (executing the instance of $\pi'_{\texttt{KA}}$ with the emulator $\mathcal{E}$). Due to Proposition 3.2, π_{1-c} gives no information on k_{1-c} to the eavesdropper $\mathcal{A}$. Therefore $\bar{m}_{1-c}$ gives no additional information to $\mathcal{Z}$. Moreover, since the execution of $\mathcal{E}$ only depends on the outputs of the dummy channel $\mathcal{F}_{\texttt{DC}}$, its view provides $\mathcal{Z}$ with no additional information, even given the rest of $\mathcal{Z}$'s view. The view of $\mathcal{Z}$ is therefore the same in both scenarios. △

5 OT protocol secure against a malicious receiver

In this section, we make our protocol secure against a malicious receiver. Note that in our semi-honest protocol, we rely on the fact that the players will engage in two instances of an emulatable key agreement protocol, where the receiver will play the role of Bob in one of them and the emulator in the other. Of course, if the receiver is malicious, he will not necessarily adopt this behaviour. We will use standard techniques to solve this problem. Namely, we want to use the paradigm introduced in [GMW86]: we will make the receiver prove in zero knowledge that he is acting as in the semi-honest protocol.

To do this, we will need that the receiver can commit to bits. Recall that in [KMS15] it was shown that commitments where the sender commits follow from any non-trivial EC, but since an EC is asymmetric, it is not clear that this allows the receiver to commit. Therefore, we solve this problem first.

5.1 Receiver commitment from any non-trivial EC

The solution in a nutshell is to observe that the commitment protocol from [DKS99] will work for receiver commitments on any non-trivial EC, if we slightly tune some of the parameters.

First, note that we can reverse the direction of the EC, by simply having the sender send a random bit x on the EC, the receiver chooses a bit b to *send* and sends $x \oplus b$ back on the public channel. This is clearly a noisy channel in the opposite direction. In this subsection we will rename the sender and call him the verifier V, while the receiver will be called the committer C. What we just constructed is a "reversed EC" where the C sends and V receives. V always receives with noise rate δ, but C can reduce his noise rate to γ if he is corrupted (and hence get a better idea of what V received). The goal is now to build an unconditionally secure commitment scheme based on such a channel.

We define some constants as follows. d_0 is defined by $\delta = \gamma(1-d_0) + d_0(1-\gamma)$. That is, d_0 is such that adding noise with rate γ and then noise with rate d_0 produces total noise rate δ. This means that $d_0 = (\delta-\gamma)/(1-2\gamma)$, and from it follows trivially that since $\delta < 1/2$, we have $\delta > d_0$. We can therefore choose constants d_1, d and d^* such that $d_0 < d_1 < d^* < d < \delta$. Finally, we define $\delta' = \gamma(1-d_1) + d_1(1-\gamma)$. Note that since $d_1 > d_0$ we have $\delta' > \delta$.

Furthermore, we define ℓ to be the logarithm of the number of elements in a Hamming ball of radius d, and likewise ℓ^* the logarithm of number of elements in a Hamming ball of radius d^*.

We will need three families of universal hash functions $\mathcal{H}, \mathcal{H}_1, \mathcal{H}_2$ that are $64k$-wise independent and map from $\{0,1\}^k$ to $\{0,1\}, \{0,1\}^{\ell^*}, \{0,1\}^{\ell-\ell^*}$, respectively.

With these definitions in place, we can use the protocols from [DKS99] with no change. We copy them here for completeness.

```
                    Protocol Commit

C:
    Sample $X \in_R \{0,1\}^k$, send $(\texttt{send}, \mathrm{sid}, C, V, X)$ to $\mathcal{F}_{\texttt{EC}}$.
V:
    Await $(\texttt{send}, \mathrm{sid}, C, V, X')$ from $\mathcal{F}_{\texttt{EC}}$
    Sample $h_1 \in_R \mathcal{H}_1$, send $(\texttt{send}, \mathrm{sid}_1, V, C, h_1)$ to $\mathcal{F}_{\texttt{Pub}}$.
C:
    Await $(\texttt{send}, \mathrm{sid}_1, V, C, h_1)$ from $\mathcal{F}_{\texttt{Pub}}$.
    Set $y_1 := h_1(X)$, send $(\texttt{send}, \mathrm{sid}_2, C, V, y_1)$ to $\mathcal{F}_{\texttt{Pub}}$.
V:
    Await $(\texttt{send}, \mathrm{sid}_2, C, V, y_1)$ from $\mathcal{F}_{\texttt{Pub}}$.
    Sample $h_2 \in_R \mathcal{H}_2$, send $(\texttt{send}, \mathrm{sid}_3, V, C, h_2)$ to $\mathcal{F}_{\texttt{Pub}}$.
C:
    Await $(\texttt{send}, \mathrm{sid}_3, V, C, h_2)$ from $\mathcal{F}_{\texttt{Pub}}$.
    Sample $h \in_R \mathcal{H}$, set $y_2 := h_2(X)$ and $b := h(X)$.
    Send $(\texttt{send}, \mathrm{sid}_4, C, V, y_2)$ and $(\texttt{send}, \mathrm{sid}_5, C, V, h)$ to $\mathcal{F}_{\texttt{Pub}}$.
    Output $b$.
V:
    Await $(\texttt{send}, \mathrm{sid}_4, C, V, y_2)$ and $(\texttt{send}, \mathrm{sid}_5, C, V, h)$ from $\mathcal{F}_{\texttt{Pub}}$.
```

Protocol Open

We define Δ as the Hamming distance.

C:

Send $(\texttt{send}, \mathrm{sid}, C, V, X)$ to $\mathcal{F}_{\texttt{Pub}}$.

V:

Await $(\texttt{sent}, \mathrm{sid}, C, V, X)$ from $\mathcal{F}_{\texttt{Pub}}$.

Check that $y_1 = h_1(X)$, $y_2 = h_2(X)$ and $\Delta(X, X') \leq \delta' k$. If either condition is false, then abort.

Output $b := h(X)$.

We have defined our constants slightly differently from what was done in [DKS99], but d_0 is defined in the same way, and the rest of the constants satisfy the same inequalities. It therefore turns out that exactly the same proofs can be used to show this version secure. We will not repeat the proofs here, but give some intuition why the protocol is secure. We let Δ denote the Hamming distance, and by negligible we mean negligible as a function of k.

Both parties are honest. In this case we expect X' to be at distance δk from X. Since $\delta' > \delta$, the probability that the distance is greater than $\delta' k$ is negligible, so V will accept the opening.

C **is corrupt.** We want to argue that there is only one string C can convincingly open after commitment time. Suppose first that C tries to claim a string X^* with $\Delta(X^*, X) > d^* k$. Then note than in his view, the received string X' is expected to be such that $\delta(X, X') = \gamma k$. So we expect that $\Delta(X^*, X') > (d^*(1-\gamma) + \gamma(1-d^*))k > \delta' k$ because $d^* > d_1$. So V would reject with overwhelming probability in this case. This means that X^* must be in a Hamming ball with radius d^* and center in X. But by sending $h_1(X), h_2(X)$, C reveals ℓ bits of information on X. Since $\ell > \ell^*$ this is more than required to identify uniquely an string in a ball of radius ℓ^*, so there is only one string that can be opened.

V **is corrupt.** We want to argue that V has essentially no information about $h(X)$ before opening. Note that in V's view X is in a Hamming ball with radius δ and center in X'. Via the hashing V gets only ℓ bits of information, and since $d < \delta$, one can show that there are exponentially many candidates left for X, even after hashing. Now by a standard privacy amplification argument, it follows that the expected information V has on $h(X)$ is negligible.

5.2 From commitment to security against malicious receiver

Recall that the GMW compiler [GMW86] transforms a semi-honestly secure protocol into a maliciously secure one by using the following three steps: in the first step, each party commits to his input; in the second step, each party is forced to commit to a random tape, where it is important that the tape is hidden from the other party and is chosen at random. This is done by having the party that is committing to a random tape commit to a random value. The other party then sends a random string. The tape is then defined to be the xor of both strings. This technique is known as coin-tossing in the well. finally, in the third step, each player follows the protocol with the committed inputs and their committed tape and whenever they send a message, they also prove in zero-knowledge that this is the correct message given their committed input, their committed random tape and the transcript of the protocol.

In this section, we are only interested in achieving security against a malicious receiver, so we apply the compiler to the receiver only. This results in the following approach: In the first step, the receiver will commit to his choice of input c; this also indicates the instance of the key agreement protocol where he will play the role of Bob. In the second step, the receiver will be forced to commit to a random tape t for the emulator using coin-tossing in the well. Then the sender and receiver will run an augmented version of the semi-honest protocol. Each instance of the key agreement protocol will be associated to an index b. Each time a receiver sends a message, the receiver also proves in zero-knowledge: "Either the given instance of key agreement has index $b = c$ or the message was produced by following the description of the emulator with random tape t".

There is, however, one difficulty: In [GMW86], the commitments were computational. It was therefore possible to prove statements about committed values directly. For a black-box information-theoretically secure commitment, it is not directly possible to prove statements that involve the committed values. To fix this problem, we use a commitment scheme which can indeed be used for any number of zero-knowledge proofs. This is the commitment scheme from [CvdGT95] which was later proven UC-secure in [Est04]. As shown in [CvdGT95], this commitment scheme can be constructed in a black-box manner from any commitment schemes. Although this commitment scheme only allows proofs of xor relationships directly, one can use use techniques such as [BCC88] to prove arbitrary statements involving the committed values.

Functionality $\mathcal{F}_{\texttt{COMZK}}$ (Commitment with zero-knowledge)

$\mathcal{F}_{\texttt{COMZK}}$ runs with two parties: a sender and a receiver.

Commit: On receiving $(\texttt{commit}, \text{cid}, m)$ from the sender:

If such a command has already been sent, ignore the message.

Otherwise, record (cid, m) and send $(\texttt{committed}, \text{cid})$ to $\mathcal{A}$ and to the receiver.

Reveal: On receiving $(\texttt{reveal}, \text{cid})$ from the sender:

If no pair (cid, m) was recorded then ignore the message. Otherwise, send $(\texttt{open}, \text{cid}, m)$ to $\mathcal{A}$ and to the receiver.

Proof: On receiving $(\texttt{prove}, x, \text{cid}_1, \dots, \text{cid}_n, R)$ from the sender:

Check that for each cid_i, there exists a m_i such that the pair (cid_i, m_i) has been recorded. If this is not the case then ignore the command. Let $w = (m_1, \dots, m_n)$. Check that $(x, w) \in R$. If this is not the case then ignore the command. Otherwise, send $(\texttt{proven}, x, \text{cid}_1, \dots, \text{cid}_n, R)$ to the receiver and $\mathcal{A}$.

Protocol $\pi_{\texttt{OTMR}}$ (Oblivious transfer–malicious receiver)

We denote b as the index of the key agreement instance. We denote m_0, m_1 as the sender's input and c denotes the receiver's input. We denote $\mathcal{E}(t, r)$, the next message function of the emulator given transcript t and random tape r. If the emulator is awaiting a message for a given transcript t, we let $\mathcal{E}(t, r) = \perp$. We define the following two relationships: R_1 and R_2.

$$R_1(a,(b,c)) := \begin{cases} 1 & a = b \oplus c \\ 0 & \text{otherwise} \end{cases}$$

$$R_2((t,m,b),(r,c)) := \begin{cases} 1 & \text{if } b = c \\ 1 & \mathcal{E}(t;r) = m \\ 0 & \text{otherwise} \end{cases}$$

Receiver:

$r_1 \in_R \{0,1\}^k$

Send $(\texttt{commit}, \mathrm{cid}, c)$, $(\texttt{commit}, \mathrm{rid}_1, r_1)$ to $\mathcal{F}_{\texttt{COMZK}}$

Sender:

Await $(\texttt{committed}, \mathrm{cid})$, $(\texttt{committed}, \mathrm{rid}_1)$ from $\mathcal{F}_{\texttt{COMZK}}$

$r_2 \in_R \{0,1\}^k$

Send r_2 to the receiver.

Receiver:

$r \leftarrow r_1 \oplus r_2$ (random tape)

Send $(\texttt{commit}, \mathrm{rid}, r)$ to $\mathcal{F}_{\texttt{COMZK}}$ (commit to the random tape)

Send $(\texttt{prove}, r_2, \mathrm{rid}_1, \mathrm{rid}, R_1)$ to $\mathcal{F}_{\texttt{COMZK}}$ (prove that the commited value associated to rid is indeed a commitment to the random tape)

Sender:

Await $(\texttt{committed}, \mathrm{rid})$ and $(\texttt{proven}, r_2, \mathrm{rid}_1, \mathrm{rid}, R_1)$ from $\mathcal{F}_{\texttt{COMZK}}$.

Sender $\leftrightarrow$ Receiver:

Sender and receiver run $\pi_{\texttt{OTSH}}$ as defined in Section 4 where the sender inputs m_0, m_1 and the receiver inputs c with the following modification:

Whenever a receiver would send a message m in the semi-honest protocol, let b be the instance of the key agreement protocol they are executing, and let t be the transcript up to that point for that instance of the key agreement protocol. The receiver sends m to the sender and also sends the command $(\texttt{prove}, (t,m,b), \mathrm{rid}, \mathrm{cid}, R_2)$ to $\mathcal{F}_{\texttt{COMZK}}$. Whenever the sender receives a message m from the the receiver, he awaits that $\mathcal{F}_{\texttt{COMZK}}$ send him $(\texttt{proven}, (t,m,b), \mathrm{rid}, \mathrm{cid}, R_2)$ before proceeding.

Theorem 5.1 *$\pi_{\texttt{OTMR}}$ securely realizes $\mathcal{F}_{\texttt{OT}}$ in the $\mathcal{F}_{\texttt{EC}}$-hybrid model against an environment that can only semi-honestly corrupt the sender.*

This theorem follows directly from the construction of XOR commitments from [CvdGT95, Est04], the security of the GMW compiler [GMW86] and the security of the zero-knowledge protocol from [BCC88, Kil92].

6 Secure protocol

In this section we consider our oblivious transfer protocol $\pi_{\mathtt{OTMR}}$ from Section 5, which is secure against a semi-honest sender and a malicious receiver and we show that, if $\pi_{\mathtt{OTMR}}$ is implemented with the key agreement protocol from Section 3.1, we can transform $\pi_{\mathtt{OTMR}}$ into a protocol $\pi_{\mathtt{OT}}$ secure against an malicious sender too.

Note that in the aforementioned key agreement protocol, the sender is supposed to send through the channel several bitstrings of length ℓ and Hamming weight either m or $m+1$, where $\ell = 2m+1$. From now on, we refer to bitstrings of weights m and $m+1$ as codewords, while the rest will be non-codewords. A problem that arises when using this key agreement protocol as a basis for our oblivious transfer protocol, is that an active sender could use non-codewords to bias the distribution of indices and learn the receiver's choice. For example, if she sends the all-one bitstring, this index will be accepted by the receiver with higher probability if he is playing the role of Bob, than it will if he is playing the role of the emulator.

We will prevent an active sender from using non-codewords in her advantage by combining cut-and-choose techniques, a typicality test and an OT-combiner. The protocol works essentially as follows: the sender and receiver will start to run N instances of $\pi_{\mathtt{OTMR}}$ in parallel. Right after the sender has sent the intended codewords through the channel $\mathcal{F}_{\mathtt{EC}}$ in all these instances, the receiver will then choose half of those instances and request the sender to open her view (i.e., to reveal the information that she sent through the channel). The receiver now runs a typicality test on those instances: he counts the number of differences between what the sender claims to have sent and what he received for those instances. If this distance is higher than what would be typically expected from the noisy channel then the receiver aborts. If the test passes then it is guaranteed that, except with negligible probability, there is at least one unopened instance where no bad codeword was sent. The sender and receiver now apply an OT combiner on the half of the instances of $\pi_{\mathtt{OTMR}}$ that have not been opened. Since the sender has behaved well in at least one of these instances, then we achieve a secure oblivious transfer protocol.

The sender could also try to cheat in the public channel part of the key agreement protocol by sending some inconsistent information (for example in the information reconciliation step) to see the aborting behaviour of the receiver; however, we have the receiver abort in the global protocol if he sees at least one inconsistency in some instance of the protocol. Given the properties of the combiner the only way to obtain information about the receiver's input bit is that the sender cheats in one of the key agreement protocols of every unopened instance and the receiver never aborts, which happens if the sender guesses each of the b_i's for the unopened instances and in turn this happens with probability $2^{-N/2}$ (in fact, we could make her cheating probability even lower by having the receiver abort if he detects inconsistent information in the opened instances).

6.1 Protocol

Before describing the protocol, we explain some details about the choice of parameters. We denote by $W(\sigma)$ the expected number of bits that the noisy channel will flip during the execution of the protocol $\pi_{\mathtt{OTMR}}$ with security parameter σ, assuming that the receiver is honest and he does not change the noise parameter of the channel. That is, if $B(\sigma)$ is the total number of bits sent through the noisy channel during the protocol, we have $W(\sigma) = \delta B(\sigma)$. Note that W is polynomial in σ.

Let κ denote the security parameter that we want to adopt in our final protocol $\pi_{\mathtt{OT}}$. We will choose the parameters N (the number of instances of $\pi_{\mathtt{OTMR}}$ that will be run), σ (the security parameter of $\pi_{\mathtt{OTMR}}$) and τ (a threshold parameter for the test, which is $W(\sigma)$ plus a small offset) so that we have the following guarantees:

1. It holds that $N \cdot 2^{-\sigma} \leq 2^{-\kappa}$. This means that the probability that at least one instance of $\pi_{\texttt{OTMR}}$ is broken is smaller than $2^{-\kappa}$.

Consider the $N/2$ instances from the testing set. Let D be the total number of bits in which the sent and received messages in these instances differ. Then

2. If the sender only sends valid codewords, the probability that D is greater than $\tau N/2$ is less than $2^{-\kappa}$.

3. If the sender sends non-valid codewords in $N - \kappa$ instances of $\pi_{\texttt{OTMR}}$, the probability that D is less than $\tau N/2$ is less than $2^{-\kappa}$.

Our oblivious transfer protocol works as follows.

Protocol $\pi_{\texttt{OT}}$ (Oblivious transfer)

The protocol involves two players: the sender and the receiver. The sender provides inputs $m_0, m_1 \in \{0,1\}$ and receives no output. The receiver provides $c \in \{0,1\}$ and outputs m_c.

We define the following parameters

$$Q(x) := \frac{32}{(1-2\delta)^2} W(x).$$
$$\sigma := \min\{x \in \mathbb{Z} : x - \log Q(x) - \log\kappa \geq \kappa\}.$$
$$N := \kappa Q(\sigma).$$
$$\tau := W(\sigma) + \frac{1-2\delta}{2}.$$

(Note that, once κ is fixed, σ is well defined because $Q(x)$ is polynomial in x and hence $x - \log Q(x) \geq \kappa + \log\kappa$ for sufficiently large x.)

Sender:

Sample $\Delta \in_R \{0,1\}$.

Sample $w_0^1, \ldots, w_0^N \in_R \{0,1\}$.

Sample $w_1^1, \ldots, w_1^N \in_R \{0,1\}$.

Let $\Delta_i := w_0^i \oplus w_1^i \oplus \Delta$, $i = 1, \ldots, N$.

Receiver:

Sample $b_1, \ldots, b_N \in_R \{0,1\}$.

Sender $\leftrightarrow$ Receiver:

Sender and receiver run N instances of the protocol $\pi_{\texttt{OTMR}}$ as defined in Section 5. Let (w_0^i, w_1^i) be the sender's input and b_i be the receiver's input in the ith instance. If at some point in one of the instances the sender sends any information through the public channel that the receiver detects as invalid (such as incorrectly formed h_1, h_2, or a value v that is not of the form $h_1(X^s)$ for any string X^s), then the receiver waits until all instances are completed and then aborts.

Moreover, the sender records the bits that she sends through the elastic channel in each of the instances as $X = \{(i, j, x_{i,j}) \mid 1 \leq i \leq N,\ 1 \leq j \leq B\}$. The receiver records the noisy version of bits that he receives from each instance as $Y = \{(i, j, y_{i,j}) \mid 1 \leq i \leq N, 1 \leq j \leq B\}$.

Receiver:

Choose $\mathcal{T} \in_R \{I \mid I \subseteq \{1, \ldots, N\}, |I| = N/2\}$.

Send $\mathcal{T}$ to receiver.

Set $\mathcal{L} := \{1, \ldots, N\} \setminus \mathcal{T}$.

Sender:

Await $\mathcal{T}$.

If $|\mathcal{T}| \neq N/2$ then abort.

Set $\mathcal{L} := \{1, \ldots, N\} \setminus \mathcal{T}$, send $S := \{(i, \Delta_i) \mid i \in \mathcal{L}\}$ and $\tilde{X} := \{(i, j, x_{i,j}) \in X \mid i \in \mathcal{T}\}$to the receiver.

Receiver:

Await $\tilde{X}$ and S.

Check that $\tilde{X}$ indeed corresponds to a set of bits that the sender should have sent in $\pi_{\texttt{OTMR}}$ (i.e., that the appropriate parts of $\tilde{X}$ correspond to codewords). If not, abort.

Check that

$$\sum_{i \in \mathcal{T}, 1 \leq j \leq B} |x_{i,j} - y_{i,j}| \leq \frac{\tau N}{2}.$$

If it fails, then abort.

Let $b := \bigoplus_{i \in \mathcal{L}} b_i$. Send $d := b \oplus c$ to the sender.

Sender:

Let $w_0 := \bigoplus_{i \in \mathcal{L}} w_0^i$, $w_1 := w_0 \oplus \Delta$. Send $(v_0, v_1) := (m_0 \oplus w_d, m_1 \oplus w_{1 \oplus d})$ to the receiver.

Receiver:

Let $w := \bigoplus_{i \in \mathcal{L}} w_{b_i}^i \oplus (b_i \wedge \Delta_i)$. Output $w \oplus v_c$.

6.2 Remarks about cut-and-choose and typicality test in $\pi_{\texttt{OT}}$

Before we prove that the protocol $\pi_{\texttt{OT}}$ realizes $\mathcal{F}_{\texttt{OT}}$ securely, we need to make some observations about the role played by our cut-and-choose technique and the typicality tests.

First of all, we note that for the sender to break security of the protocol, she must break each instance in the evaluation set $\mathcal{L}$, which is decided by the receiver. If, for more than κ of the N total instances of $\pi_{\texttt{OTMR}}$, the corrupt sender sends exclusively *correct* codewords then, except with probability $2^{-\kappa}$, at least one of those instances will end up in $\mathcal{L}$ and the receiver would be protected in that case. Therefore, we will assume that the testing set $\mathcal{T}$ contains at least $\frac{N}{2} - \kappa$ instances where the receiver sent a bad codeword.

Define $\mathsf{X}[\mu]$ to be a binomial variable with expectation μ. By abuse of notation, we denote by $\sum_{i=1}^{N} \mathsf{X}[\mu]$ the variable defined by sampling N independent random variables with expectation μ and adding the result.

6.2.1 Probability that an honest sender passes typicality test.

We show that the honest sender passes the typicality test with probability at least $1 - 2^{-\kappa}$. Let $T = \sum_{i=1}^{N/2} \mathsf{X}[W(\sigma)]$. An honest receiver does *not* pass the typicality test if and only if $T \geq \tau N/2$. Now let $\mu = \mathsf{E}[T] = \frac{N}{2}W(\sigma)$ and $\beta = \frac{1}{2W(\sigma)}$. We can apply Chernoff's bound to see that

$$\Pr[T \geq \tau N/2] = \Pr[T \geq (1+\beta)\mu] \leq e^{-\mu\beta^2/4} \leq 2^{-\kappa}.$$

6.2.2 Probability that a malicious sender breaks the typicality test.

We show that if a malicious sender cheats in $N/2 - \kappa$ instances of the testing set, she passes the typicality test with probability at most $2^{-\kappa}$. Note that in order for the sender to send something different from a codeword in a given instance, at least one of the bits she sent does not correspond to the bit she communicates when she sends $\tilde{X}$. Now note that, for a given bit $x_{i,j}$ communicated by the sender when she sends $\tilde{X}$, if this bit was indeed correct, then $x_{i,j} \neq y_{i,j}$ with probability δ, while if she sent $1 - x_{i,j}$ instead, then $x_{i,j} \neq y_{i,j}$ with probability $1 - \delta$. Note that the difference between these probabilities is $1 - 2\delta$. This means that, in expectation, if the sender assumes the cheating behaviour we just described, the distance between the bitstrings $(x_{i,j})$ and $(y_{i,j})$ will grow by an additive factor of $(1 - 2\delta)(N/2 - \kappa)$ with respect to the case where the sender would be honest. We want to show that in these conditions, the malicious sender will fail the test with high probability. That is, again defining $T = \sum_{i=1}^{N/2} \mathsf{X}[W(\sigma)]$, we need to show:

$$\Pr\left[T \leq \frac{\tau N}{2} - (1-2\delta)\left(\frac{N}{2} - \kappa\right)\right] \leq 2^{-\kappa}.$$

Let $\mu = \mathsf{E}[T] = \frac{N}{2}W(\sigma)$ and $\beta = \frac{(1-2\delta)(N-4\kappa)}{2NW(\sigma)}$. Chernoff's bound then says

$$\Pr[T \leq (1-\beta)\mu] \leq e^{-\mu\beta^2/2}.$$

Now it is easy to see that, for the values of μ and β detailed above, $(1-\beta)\mu = \frac{\tau N}{2} - (1 - 2\delta)\left(\frac{N}{2} - \kappa\right)$ (so this probability is indeed what we want to bound) and that $e^{-\mu\beta^2/2} \leq 2^{-\kappa}$.

In the rest of the section we will prove the following statement.

Theorem 6.1 *π_{OT} securely realizes $\mathcal{F}_{\mathrm{OT}}$ in $\mathcal{F}_{\mathrm{EC}}$-hybrid model.*

6.3 Correctness

If both players are honest, then the protocol is correct with probability at least $1 - 2^{-\kappa}$. Indeed, with at least this probability the honest sender passes the typicality test and the protocol is completed. Then, note that:

$$w^i_{b_i} \oplus (b_i \wedge \Delta_i) = \begin{cases} w^i_0 & \text{, if } b_i = 0 \\ w^i_1 \oplus \Delta_i = w^i_0 \oplus \Delta & \text{, if } b_i = 1 \end{cases}$$

Hence $w = w_0$ if there is an even number of $i \in \mathcal{L}$ such that $b_i = 1$, i.e., if $b = 0$, and $w = w_0 \oplus \Delta = w_1$ if $b = 1$. In other words, $w = w_b$.

On the other hand $v_c = m_c \oplus w_{c \oplus d} = m_c \oplus w_b$.

Therefore the output of the receiver equals $w \oplus v_c = m_c$, so the protocol outputs the correct value.

6.4 Security against a malicious receiver

Simulation. The simulator $\mathcal{S}$ for $\pi_{\texttt{OT}}$ will first proceed by running N instances $\mathcal{S}_1, \ldots, \mathcal{S}_N$ of the simulator for $\pi_{\texttt{OTMR}}$. Upon receiving $(\texttt{choice}, b_i)$ from the environment, it will record it and send a random w_i. If any of the simulators aborts, then the simulator aborts.

In the next step, it awaits the test set $\mathcal{T}$ from $\mathcal{Z}$. Now, the simulator must send a $\tilde{X}$ such that the view of $\mathcal{Z}$ for the test instances is the same as in the real world.

Each of the views produced by the simulators are statistically indistinguishable (within $2^{-\sigma}$) from real instances of the OT protocol.

Therefore, there must be a distribution D for $\tilde{X}$ that depends only on the transcript between the simulators and $\mathcal{Z}$ that is $(1/2^{\kappa})$-close to one which would be produced in the real world.

Indeed, if this was not the case, since

$$\frac{N}{2^{\sigma}} = \frac{\kappa Q(\sigma)}{2^{\sigma}} = \frac{1}{2^{\sigma - \log Q(\sigma) - \log \kappa}} \leq \frac{1}{2^{\kappa}},$$

then $\mathcal{Z}$ would be able to distinguish with probability larger than $1/2^{\sigma}$ between a run of the simulated malicious-receiver OT and a run with the malicious-receiver OT protocol with the elastic channel for at least one of the N instances, which contradicts the security of $\pi_{\texttt{OTMR}}$.

$\mathcal{S}$ samples $\tilde{X} \in_R D$. $\mathcal{S}$ sets $\mathcal{L} = \{1, \ldots, N\} \setminus \mathcal{T}$. $\mathcal{S}$ samples $\Delta_i \in_R \{0,1\}$, for $i \in \mathcal{L}$ and sets $S = \{(i, \Delta_i) \mid i \in \mathcal{L}\}$.

$\mathcal{S}$ sends $S, \tilde{X}$ to $\mathcal{Z}$. $\mathcal{S}$ computes $w := \bigoplus_{i \in \mathcal{L}} w_i \oplus (\Delta_i \wedge b_i)$ and $b := \bigoplus_{i \in \mathcal{L}} b_i$. $\mathcal{S}$ awaits that the environment inputs d. $\mathcal{S}$ samples a random $x \in_R \{0,1\}$, sets $c = b \oplus d$ and sends $(\texttt{choice}, c)$ to $\mathcal{F}_{\texttt{OT}}$. Upon receiving $(\texttt{receipt}, m)$, $\mathcal{S}$ sets $u_0 = m \oplus w, u_1 = x$ and sends $(v_0, v_1) := (u_d, u_{1 \oplus d})$ to $\mathcal{Z}$.

Indistinguishability. This follows directly from the fact that the given robust OT-combiner is universally composable and that the underlying OT protocol is secure against a malicious receiver.

6.5 Security against a malicious sender

Simulation. The simulator $\mathcal{S}$ employs the following strategy. First, for each instance of OT, $\mathcal{S}$ runs an instance $\mathcal{S}_i$ of the simulator for the protocol $\pi_{\texttt{OTMR}}$ (for the semi-honest sender) for as long as $\mathcal{Z}$ does not send invalid codewords for that instance. If any $\mathcal{S}_i$ aborts, then $\mathcal{S}$ aborts.

When for a given instance, $\mathcal{Z}$ sends an invalid codeword, $\mathcal{S}$ takes the simulator $\mathcal{S}_i$, samples a b_i at random and samples a receiver R_i whose input is b_i and whose view is consistent with what has been sent by the environment for the given instance.

From this point on, instead of running the simulator for the given instance, $\mathcal{S}$ runs R_i and whenever $\mathcal{Z}$ sends a message which is meant to be communicated through the elastic channel, $\mathcal{S}$ simulates the channel and sends the result to R_i.

Once the instances of OT (both simulated and run with honest receiver) have completed, $\mathcal{S}$ samples a random test set $\mathcal{T}$ and sends it to $\mathcal{Z}$. $\mathcal{S}$ awaits $\tilde{X}, S$ from the environment. $\mathcal{S}$ simulates the typicality test. $\mathcal{S}$ takes each instance of OT for the test that is still run by the simulator for the test cases and replace it with a receiver in the same way that was described above. Then once $\mathcal{S}$ has produced the given views, $\mathcal{S}$ takes these views and runs the typicality test. If the test fails, the simulator aborts.

$\mathcal{S}$ denotes the set of instances $\mathcal{I}$ that were only run by simulators and were not part of the test set. Let $\mathcal{J}$ be the set of instances that were run by the receivers and were not part of the test set. The simulators provided the values $\{(w_0^i, w_1^i) \mid i \in \mathcal{I}\}$ and the receivers provided the values $\{w_{b_j}^j \mid j \in \mathcal{J}\}$.

$\mathcal{S}$ samples a $u \in \mathcal{I}$ and, for each $i \in \mathcal{I}$, selects a random b_i. $\mathcal{S}$ selects $b = \bigoplus_{i \in \mathcal{L}, i \neq u} b_i$, $w := \bigoplus_{i \in \mathcal{L}, i \neq u} w_{b_i}^i \oplus (\Delta_i \wedge b_i)$. $\mathcal{S}$ sets $m_0' := w \oplus w_0^u$, $m_1' := w \oplus w_1^u \oplus \Delta_u$. $\mathcal{S}$ samples a random r and sends $d = b \oplus r$ to $\mathcal{Z}$. $\mathcal{S}$ awaits v_0, v_1 from $\mathcal{Z}$. $\mathcal{S}$ sets $m_0 := w \oplus v_{b \oplus r} \oplus m'_{b \oplus r}$ and $m_1 := w \oplus v_{b \oplus r \oplus 1} \oplus m'_{b \oplus r \oplus 1}$. $\mathcal{S}$ sends $(\mathtt{send}, \mathrm{sid}, m_0, m_1)$ to $\mathcal{F}_{\mathtt{OT}}$.

Indistinguishability. The real-world instances of OT where the sender did not send bad codewords are indistinguishable from the instances run by local simulators in the ideal world. This follows directly from the security of $\pi_{\mathtt{OTSH}}$ against semi-honest adversaries.

Next, we consider, the instances of OT where the sender sent bad codewords. These are also indistinguishable from instances run by the simulator because, on seeing a bad codeword, the simulator replaces the local simulator with a receiver R_i, with random input b_i, that acts as in the real world (including the communication between the sender and this receiver, which is simulated by imitating the behaviour of the channel). Furthermore, the receiver is constructed so that it is consistent with what had been previously sent through the channel and the given choice of inputs.

The last step of our simulation needs, however, to make sure that $\mathcal{I}$ is non-empty, i.e., that there is at least one instance of the evaluation set where $\mathcal{Z}$ sends only correct codewords. But notice that, as we have shown before, if $\mathcal{Z}$ would send a non-codeword in each instance, it would result (except with probability $2^{-\kappa}$) in an abort due to the typicality test.

References

[BBCM95] Charles H Bennett, Gilles Brassard, Claude Crépeau, and Ueli M Maurer. Generalized privacy amplification. *Information Theory, IEEE Transactions on*, 41(6):1915–1923, 1995.

[BBR88] Charles H Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM journal on Computing*, 17(2):210–229, 1988.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCS96] Gilles Brassard, Claude Crépeau, and Miklos Santha. Oblivious transfers and intersecting codes. *Information Theory, IEEE Transactions on*, 42(6):1769–1780, 1996.

[BS94] Gilles Brassard and Louis Salvail. Secret-key reconciliation by public discussion. In *advances in Cryptology—EUROCRYPT'93*, pages 410–423. Springer, 1994.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.

[CK88] Claude Crépeau and Joe Kilian. Achieving Oblivious Transfer Using Weakened Security Assumptions (Extended Abstract). In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 42–52, 1988.

[CM97] Christian Cachin and Ueli M Maurer. Linking information reconciliation and privacy amplification. *journal of Cryptology*, 10(2):97–110, 1997.

[CMW05] Claude Crépeau, Kirill Morozov, and Stefan Wolf. Efficient unconditional oblivious transfer from almost any noisy channel. In *Security in Communication Networks*, pages 47–59. Springer, 2005.

[Cré97] Claude Crépeau. Efficient Cryptographic Protocols Based on Noisy Channels. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 306–317, 1997.

[CS06] Claude Crépeau and George Savvides. Optimal Reductions Between Oblivious Transfers Using Interactive Hashing. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 201–221, 2006.

[CvdGT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed Oblivious Transfer and Private Multi-Party Computation. In *Advances in Cryptology - CRYPTO 95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, pages 110–123, 1995.

[DFMS04] Ivan Damgård, Serge Fehr, Kirill Morozov, and Louis Salvail. *Unfair noisy channels and oblivious transfer.* Springer, 2004.

[DKS99] Ivan Damgård, Joe Kilian, and Louis Salvail. On the (Im)possibility of Basing Oblivious Transfer and Bit Commitment on Weakened Security Assumptions. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 56–73, 1999.

[Est04] Gregory Estren. *Universally composable committed oblivious transfer and multiparty computation assuming only basic black-box primitives.* PhD thesis, McGill University, 2004.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design. In *Advances in Cryptology—CRYPTO'86*, pages 171–185. Springer, 1986.

[HR11] Thomas Holenstein and Renato Renner. On the randomness of independent experiments. *Information Theory, IEEE Transactions on*, 57(4):1865–1871, 2011.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and Jürg Wullschleger. Constant-rate oblivious transfer from noisy channels. In *Advances in Cryptology–CRYPTO 2011*, pages 667–684. Springer, 2011.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732. ACM, 1992.

[KMS15] Dakshita Khurana, Hemanta K Maji, and Amit Sahai. Secure Computation from Elastic Noisy Channels. *Accepted at Eurocrypt 2016.* `http://web.cs.ucla.edu/~dakshita/Dakshita\_Khurana\_Home\_files/KMS15.pdf`, 2015. Accessed: 2016-02-09.

[Mau93] Ueli M. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39(3):733–742, 1993.

[MW00] Ueli Maurer and Stefan Wolf. Information-theoretic key agreement: From weak to strong secrecy for free. In *Advances in CryptologyEUROCRYPT 2000*, pages 351–368. Springer, 2000.

[PDMN11] Adriana C. B. Pinto, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. Achieving Oblivious Transfer Capacity of Generalized Erasure Channels in the Malicious Model. *IEEE Transactions on Information Theory*, 57(8):5566–5571, 2011.

[RW05] Renato Renner and Stefan Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In *Advances in cryptology-ASIACRYPT 2005*, pages 199–216. Springer, 2005.

A Universal composability

Universal Composability is based on the simulation paradigm. Roughly, the idea is to compare the execution of the actual protocol (the real world) with an idealized scenario (the ideal world) in which the computations are carried out by a trusted third party (the ideal functionality) which receives inputs from and hands in outputs to the players. The goal is to show that these two worlds are indistinguishable. In order to formalize this goal, we introduce a party called the environment $\mathcal{Z}$, whose task is to distinguish between both worlds. Furthermore, in the ideal world, we introduce a simulator $\mathcal{S}$, its task being to simulate any action of the adversary in the real protocol and thereby to make the two views indistinguishable for any environment. More precisely, in the real world execution of protocol π, with the adversary $\mathcal{A}$ and environment $\mathcal{Z}$, the environment provides input and receives output from both $\mathcal{A}$ and π. Call $\texttt{Real}_{\mathcal{A},\pi,\mathcal{Z}}$ the view of $\mathcal{Z}$ in this execution. In the ideal world $\mathcal{Z}$ provides input and receives output from $\mathcal{S}$ and the ideal functionality $\mathcal{F}$. Call $\texttt{Ideal}_{\mathcal{S},\mathcal{F},\mathcal{Z}}$ the view of $\mathcal{Z}$ in the ideal execution. We can proceed to define what it means for a protocol to be secure.

Definition A.1 *A protocol π UC-implements a functionality $\mathcal{F}$ against a certain class of adversaries $\mathcal{C}$ if for every adversary $\mathcal{A} \in \mathcal{C}$ there exists a simulator $\mathcal{S}$ such that for every environment $\mathcal{Z}$, $\texttt{Real}_{\mathcal{A},\pi,\mathcal{Z}} \approx \texttt{Ideal}_{\mathcal{S},\mathcal{F},\mathcal{Z}}$.*

The cornerstone of the universal composability framework is the composition theorem, which works as follows. Denote by $\pi \diamond G$ a protocol π that during its execution makes calls to an ideal functionality G. The composition proof shows that if $\pi_f \diamond G$ securely implements $\mathcal{F}$ and if π_g securely implements G then $\pi_f \diamond \pi_g$ securely implements $\mathcal{F}$. This provides modularity in construction of protocols and simplifies proofs dramatically. It is also shown that proving security against a dummy adversary, one who acts as a communication channel, is sufficient for proving general security.

B Proof of Lemma 3.3

First, note that obviously $\Pr[Z \in S_0 | X = 0] \geq 1/2$ since $\gamma < 1/2$. Let $\overline{X}$ denote the random variable describing the encoding of $b = 0$ by Alice, i.e., $\overline{X}$ has the uniform distribution over the set of bitstrings in $\{0,1\}^\ell$ of weight exactly m or $m+1$. We observe that since Eve's noise is independent and identically distributed for each bit sent through the wiretap channel, for every string $\overline{x} \in \{0,1\}^\ell$ of weight m, $\Pr[Z \in S_0 | X = 0] = \Pr[Z \in S_0 | \overline{X} = \overline{x}]$. So we now compute $\Pr[Z \in S_0 | \overline{X} = \overline{x}]$ for $\overline{x} = 010101\ldots010$.

For $i = 1, \ldots, m$, let V_i be the random variable that takes value 1 if $Z = z$ and $z_{2i-1} = z_{2i} = 1$, the value -1 if $z_{2i-1} = z_{2i} = 0$ and the value 0 if $z_{2i-1} \neq z_{2i}$. Then clearly $\Pr[Z \in S_0 | \overline{X} = \overline{x}] \leq \Pr[\sum_{i=1}^m V_i \leq 0]$.

Now note that V_i are independent identically distributed variables such that $\Pr[V_i = 1] = \Pr[V_i = -1] = p$ and $\Pr[V_i = 0] = 1 - 2p$ where $p = \gamma(1-\gamma)$. Hence $\Pr[\sum_{i=1}^m V_i < 0] = \Pr[\sum_{i=1}^m V_i > 0]$ and clearly (using for example the central limit theorem) $\Pr[\sum_{i=1}^m V_i = 0] \to 0$ as ℓ (and consequently m) grows. Therefore

$$1/2 \leq \Pr[Z \in S_0 | X = 0] \leq \Pr[\sum_{i=1}^{m} V_i \leq 0] \to 1/2$$

and consequently $\Pr[Z \in S_0 | X = 0] \to 1/2$.