# On Black-Box Complexity of Universally Composable Security in the CRS model

Carmit Hazay* Muthuramakrishnan Venkitasubramaniam†

**Abstract**

In this work, we study the intrinsic complexity of *black-box Universally Composable (UC) secure computation* based on *general assumptions*. We present a thorough study in various corruption modelings while focusing on achieving security in the common reference string (CRS) model. Our results involve the following:

- **Static UC secure computation.** Designing *the first* static UC secure oblivious transfer protocol based on public-key encryption and stand-alone semi-honest oblivious transfer. As a corollary we obtain the first black-box constructions of UC secure computation assuming only two-round semi-honest oblivious transfer.
- **One-sided UC secure computation.** Designing adaptive UC secure two-party computation with single corruptions assuming public-key encryption with oblivious ciphertext generation.
- **Adaptive UC secure computation.** Designing adaptively secure UC commitment scheme assuming only public-key encryption with oblivious ciphertext generation. As a corollary we obtain the first black-box constructions of adaptive UC secure computation assuming only (trapdoor) simulatable public-key encryption (as well as a variety of concrete assumptions).

  We remark that such a result was not known even under non-black-box constructions.

**Keywords:** UC Secure Computation, Black-Box Constructions, Oblivious Transfer, UC Commitments

*Faculty of Engineering, Bar-Ilan University, Israel. Email: carmit.hazay@biu.ac.il.

†University of Rochester, Rochester, NY 14611, NY. Email: muthuv@cs.rochester.edu.

# 1 Introduction

Secure multi-party computation enables a set parties to mutually run a protocol that computes some function $f$ on their private inputs, while preserving a number of security properties. Two of the most important properties are privacy and correctness. The former implies data confidentiality, namely, nothing leaks by the protocol execution but the computed output. The later requirement implies that no corrupted party or parties can cause the output to deviate from the specified function. It is by now well known how to securely compute any efficient functionality [Yao86, GMW87, MR91, Bea91, Can01] in various models and under the stringent simulation-based definitions (following the ideal/real paradigm). Security is typically proven with respect to two adversarial models: the semi-honest model (where the adversary follows the instructions of the protocol but tries to learn more than it should from the protocol transcript), and the malicious model (where the adversary follows an arbitrary polynomial-time strategy), and feasibility results are known in the presence of both types of attacks. The initial model considered for secure computation was of a static adversary where the adversary controls a subset of the parties (who are called corrupted) before the protocol begins, and this subset cannot change. In a stronger corruption model the adversary is allowed to choose which parties to corrupt throughout the protocol execution, and as a function of its view; such an adversary is called adaptive.

These feasibility results rely in most cases on stand-alone security, where a *single* set of parties run a *single* execution of the protocol. Moreover, the security of most cryptographic protocols proven in the stand-alone setting does not remain intact if many instances of the protocol are executed concurrently [Lin03]. The strongest (but also the most realistic) setting for concurrent security is known by *Universally Composable* (UC) security [Can01]. This setting considers the execution of an unbounded number of concurrent protocols in an arbitrary and adversarially controlled network environment. Unfortunately, stand-alone secure protocols typically fail to remain secure in the UC setting. In fact, without assuming some *trusted help*, UC security is impossible to achieve for most tasks [CF01, CKL06, Lin03]. Consequently, UC secure protocols have been constructed under various *trusted setup* assumptions in a long series of works; see [BCNP04, CDPW06, KLP07, CPS07, LPV09, DMRV13] for few examples.

In this work, we are interested in understanding the intrinsic complexity of *UC secure computation*. Identifying the general assumptions required for a particular cryptographic task provides an abstraction of the functionality and the specific hardness that is exploited to obtain a secure realization of the task. The expressive nature of general assumptions allows the use of a large number of concrete assumptions of our choice, even one that may not have been considered at the time of designing the protocols. Constructions that are based on general assumptions are proven in two flavors:

**Black-box usage:** A construction is black-box if it refers only to the input/output behavior of the underlying primitives.

**Non-black-box usage:** A construction is non-black box if it uses the code computing the functionality of the underlying primitives.

Typically, non-black-box constructions have been employed to demonstrate feasibility and derive the minimal assumptions required to achieve cryptographic tasks. An important theoretical question is whether or not non-black-box usage of the underlying primitive is necessary in a construction. Besides its theoretical importance, obtaining black-box constructions is related to efficiency as an undesirable effect of non-black-box constructions is that they are typically inefficient and unlikely to be implemented in practice. Fortunately, a recent line of works [IKLP06, Hai08, PW09, GLOV12] has narrowed the gap between what is achievable via non-black-box and black-box constructions under minimal assumptions.

More relevant to our context, the work of Ishai, Prabhakaran and Sahai [IPS08] provided the first black-box constructions of UC secure protocols assuming only one-way functions in a model where all parties have

access to an ideal oblivious transfer (OT) functionality. Orthogonally, Choi et al. [CDMW09b] provided a compiler that transforms any semi-honest OT to a protocol that is secure against malicious static adversaries *in the stand-alone* (i.e. not UC) while assuming that all parties have access to the ideal commitment functionality. In the adaptive setting, the work of Choi et al. provides a transformation from adaptively secure semi-honest oblivious transfer to one that is secure *in the stronger UC setting* against malicious adaptive adversaries while assuming that all parties have access to the ideal commitment functionality. In essence, these works provide black-box constructions, however, they fall short of identifying the necessary minimal general computational assumptions in the UC setting.

Loosely speaking, a UC commitment scheme [CF01] is a fundamental building block in secure computation which is defined in two phases: in the commit phase a committer commits to a value while keeping it hidden, whereas in the decommit phase the committer reveals the value that it previously committed to. In addition to the standard binding and hiding security properties that any commitment must adhere, commitment schemes that are secure in the UC framework must allow straight-line extraction (where a simulator should be able to extract the content of any valid commitment generated by the adversary) and straight-line equivocation (where a simulator should be able to produce many commitments for which it can later decommit to both $0$ and $1$). We stress that even security in the static setting requires some notion of equivocation. Due to these rigorous requirements, it has been a real challenge to design black-box constructions of UC secure commitment schemes.

In the context of realizing the UC commitments in the CRS model, Damgård and Nielsen introduced the notion of mixed-commitments in [DN00]. This construction requires a CRS that is linear in the number of parties and can be instantiated under the $N$-residuosity and $p$-subgroup hardness assumptions. In the global CRS model (where a single CRS is introduced for any number of executions), the only known constructions are by Damgård and Groth [DG03] based on the Strong RSA assumption and Lindell [Lin11] based on the DDH assumption, where the former construction guarantees security in the adaptive setting whereas the later construction provides static security.

Another fundamental building block in secure computation which has been widely studied is oblivious transfer [Rab81, EGL85]. Semi-honest two-round oblivious transfer can be constructed based enhanced trapdoor permutations [EGL85] and smooth projective hashing [HK12], and concretely under Discrete Diffie-Hellman (DDH) [NP01]. Two-round protocols with malicious UC security are presented in the influential paper by Peikert et al. [PVW08] that presents a black-box framework in the common reference string (CRS) model for oblivious transfer, based on dual-mode public-key encryption (PKE) schemes, which can be concretely instantiated under the DDH, quadratic residuosity and Learning with Errors (LWE) hardness assumptions. In a followup work [CKWZ13], the authors present UC oblivious transfer constructions in the global CRS model assuming DDH, $N$-residuosity and the Decision Linear Assumption (DLIN). As pointed out in [CKWZ13], the [PVW08] constructions require a distinct CRS per party. In the context of adaptive UC oblivious transfer protocols, the works of [CDMW09b] and [GWZ09] give constructions in the UC commitment hybrid model where they additionally rely on an assumption that implies adaptive semi-honest oblivious transfer.

It is worth noting that while the works of [PVW08] and [CKWZ13] provide abstractions of their assumptions, the assumptions themselves are not general enough to help understand the minimal assumptions required to achieve static UC security. In particular, when restricting attention to black-box constructions based on general assumptions, the state-of-the-art literature seems to indicate that achieving UC security in most trusted setup models reduces to constructing two apparently incomparable primitives: *semi-honest oblivious transfer* and *UC commitment schemes*. This leaves the following important question open:

> *What are the minimal (general) assumptions required to construct UC secure protocols, given only* black-box *access to the underlying primitives?*

We note that this question is already well understood in the static setting when relaxing the black-box

requirement. Namely, in [DNO10] Damgård, Nielsen and Orlandi showed how to construct UC commitments assuming only semi-honest oblivious transfer in the global CRS model, while additionally assuming a pre-processing phase where the parties participate in a round-robin manner.[1] More recently, Lin, Pass and Venkitasubramaniam [LPV12] improved this result by removing any restricted pre-processing phase. In the same work the authors showed how to achieve UC security in the global CRS model assuming only the existence of semi-honest oblivious transfer. In particular, this construction shows that static UC security can be achieved without assuming UC commitments when relaying on non-black-box techniques.

In the stand-alone (i.e. not UC) setting, assuming only the existence of semi-honest oblivious transfer [Hai08, IKLP06, HIK$^+$11] show how to construct secure multiparty computation protocols while relying on the underlying primitives in a black-box manner. More recently, [CDMW09b] provided black-box constructions that are secure against static adversaries, again, in the stand-alone setting, where all parties have access to an ideal commitment functionality (cf. Proposition 1 in [CDMW09b]). The latter construction achieves a stronger notion of straight-line simulation, however falls short of achieving static UC-security (see more details in Section 3).

In the adaptive setting, the only work that considers a single general assumption that implies adaptive UC security using non-black-box techniques is the result due to Dachman-Soled et al. [DMRV13], that shows how to obtain adaptive UC commitments assuming simulatable PKE. Moreover, the best known general assumptions required to achieve black-box UC security are adaptive semi-honest oblivious transfer and UC commitments [DN02, CDMW09b]. Known minimal general assumptions that are required to construct these primitives are (trapdoor) simulatable PKE for adaptive semi-honest oblivious transfer [CDMW09a] and mixed commitments for UC commitments [DN02].

## 1.1 Our Results

In this paper we present a thorough study of black-box UC secure computation in the CRS model; details follow.

### 1.1.1 Static UC Secure Computation

Our first result is given in the static setting, where we demonstrate the feasibility of UC secure computation based on semi-honest oblivious transfer and extractable commitments. More concretely, we prove how to transform any statically semi-honest secure oblivious transfer into one that is secure in the presence of malicious adversaries, giving *only black-box access* to the underlying semi-honest oblivious transfer protocol. Our approach is inspired by the protocols from [HIK$^+$11] and [LP12], where we observe that it is not required to use the full power of static UC commitments. Instead, we employ a weaker primitive that only requires straight-line input extractability. Interestingly, we prove that this weaker notion of security, denoted by extractable commitments [MPR10], can be realized based on any CPA secure PKE. More precisely, we prove the following theorem.

**Theorem 1.1.** *(Informally) Assuming the existence of PKE and semi-honest oblivious transfer, then any functionality can be realized in the CRS model with static UC security, where the underlying primitives are accessed in a black-box manner.*

We remark here that this theorem makes a significant progress towards reducing the general assumptions required to construct UC secure protocols. Previously, the only general assumptions based on which we knew how to construct UC secure protocols were mixed-commitments [DN00] and dual-mode PKE [PVW08] both of which were tailor-made for the particular application. Towards understanding the required minimal

[1] In such a pre-processing phase, it is assumed that at most one party is allowed to transmit messages in any round.

assumptions, we recall the work Damgård and Groth in [DG03] who showed that the existence of UC commitments in the CRS model implies a stand-alone key agreement protocol. Moreover, under black-box constructions, the seminal work of Impagliazzo and Rudich [IR88] implies that key agreement cannot be based on one-way functions. Thus, there is reasonable evidence to believe that some public-key primitive is required for UC commitments. In that sense, our assumption regarding PKE is close to being optimal. Nevertheless, it is unknown whether the semi-honest oblivious transfer assumption is required.

Our result is shown in two phases. At first we compile the semi-honest oblivious transfer protocol into a new protocol with intermediate security properties in the presence of malicious adversaries. This transformation is an extension of the [HIK$^+$11] transformation that is only proven for bit oblivious transfer, whereas our proof works for string oblivious transfer. Next, we use the transformed oblivious transfer protocol in order to construct a maliciously fully secure oblivious transfer. By combining our oblivious transfer protocol with the [IPS08] protocol we obtain a statically generic UC secure computation.

An important corollary is deduced from the work by Gertner et al. [GKM$^+$00], who provided a black-box construction of PKE based on any two-round semi-honest oblivious transfer protocol. Specifically, the combination of their result with ours implies the following corollary, which demonstrates that two-round semi-honest oblivious transfer is sufficient in the CRS model to achieve black-box constructions of UC-secure protocols. Namely,

**Corollary 1.2.** *(Informally) Assuming the existence of two-round semi-honest oblivious transfer, then any functionality can be UC realized in the CRS model, where the oblivious transfer is accessed in a black-box manner.*

**Implications.** In what follows, we make a sequence of interesting observations that are implied by our result in the static UC setting.

- The important result by Canetti, Lindell, Ostrovsky and Sahai [CLOS02] presents the first *non-black-box* constructions of static UC secure protocols assuming enhanced trapdoor permutations. In fact, their result can be extended assuming only PKE with oblivious ciphertext generation (which is PKE with the special property that a ciphertext can be obliviously sampled without the knowledge of the plaintext, and can be further realized using enhanced trapdoor permutation). In that sense, our result, assuming PKE with oblivious ciphertext generation, can be viewed as an improvement of [CLOS02] when relying on this primitive in a *black-box* manner.

- The pair of works by Damgard, Nielsen and Orlandi [DNO10] and Lin, Pass and Venkitasubramaniam [LPV12] demonstrate that *non-black-box* constructions of UC commitments, and more generally static UC secure computation, can be achieved in the CRS model assuming only semi-honest oblivious transfer. In comparison, our result shows that two-round semi-honest oblivious transfer protocols are sufficient for obtaining *black-box* UC secure computation in the CRS model. Note that most semi-honest oblivious transfer protocols anyway require only two-round of communication, e.g., [EGL85].

- In [LPV09, LPV12], Lin, Pass and Venkitasubramaniam provided a unified framework for constructing UC secure protocols in any "trusted-setup" model. Their result is achieved by capturing the minimal requirement that implies UC computations in the setup model. More precisely, they introduced the notion of a UC puzzle and showed that any setup model that admits a UC puzzle can be used to securely realize any functionality in the UC setting, while additionally assuming the existence of semi-honest oblivious transfer. Moreover, they showed how to easily construct such puzzles in most models. We remark that our approach can be viewed as providing a framework to construct black-box UC secure protocols in other UC models. More precisely, we show that any setup model that admits the extractable commitment functionality can be used to securely realize any functionality assuming

the existence of semi-honest oblivious transfer. In fact, our result easily extends to the chosen key registration authority (KRA) model [BCNP04], where it is assumed the existence of a trusted authority that samples public key, secret key pairs for each party, and broadcasts the public key to all parties. We leave it for future work to instantiate our framework in other setup models.

- The fact that our construction only requires PKE and semi-honest oblivious transfer allows an easy translation of static UC security to various efficient implementations under a wider range of concrete assumptions. Specifically, both PKE and (two-round) semi-honest oblivious transfer can be realized under RSA, LWE, factoring Blum integers, DDH, $N$-residuosity and $p$-subgroup. This is compared to prior results that could only be based on the later three assumptions or Strong-RSA or a combination of concrete assumptions for static UC commitment scheme (i.e., $N$-residuosity, $p$-subgroup, Strong RSA, DDH) and semi-honest oblivious transfer (i.e., DDH, LWE, factoring Blum integers, $N$-residuosity, $p$-subgroup, RSA).

- Recently, Maji, Prabhakaran, and Rosulek [MPR10] initiated the study of the cryptographic complexity of secure computation tasks, while characterizing the relative complexity of a task in the UC setting. Specifically, they established a zero-one law that states that any task is either trivial (i.e., it can be reduced to any other task), or complete (i.e., to which any task can be reduced to), where a functionality $\mathcal{F}$ is said to *reduce* to another functionality $\mathcal{G}$, if there is a UC secure protocol for $\mathcal{F}$ using ideal access to $\mathcal{G}$. More precisely, they showed that assuming the existence of semi-honest oblivious transfer, every finite two-party functionality is either trivial or complete. While their main theorem relies on the minimal assumption of semi-honest oblivious transfer, their use of the assumption is non-black-box and they leave it as an open problem to achieve the same while relying on oblivious transfer in a black-box manner. Our result makes progress towards establishing this.

  In more details, their high-level approach is to identify complete functionalities using four categories, namely, (1) $\mathcal{F}_{\mathrm{XOR}}$ that abstracts a XOR-type functionality, (2) $\mathcal{F}_{\mathrm{CC}}$ that abstracts a simple cut-and-choose functionality, (3) $\mathcal{F}_{\mathrm{OT}}$ the oblivious transfer functionality, and (4) $\mathcal{F}_{\mathrm{COM}}$ the commitment functionality. They then show that each category can be used to securely realize any computational task.[2] Among these reductions, functionalities $\mathcal{F}_{\mathrm{XOR}}$ and $\mathcal{F}_{\mathrm{CC}}$ rely on oblivious transfer in a non-black-box way. In this work we improve the reduction of functionality $\mathcal{F}_{\mathrm{CC}}$. That is, we obtain this improvement by showing that the extractable commitment functionality $\mathcal{F}_{\mathrm{EXTCOM}}$ and semi-honest oblivious transfer can be used in a black-box way to realize functionality $\mathcal{F}_{\mathrm{OT}}$, and combine this with a reduction presented in [MPR10] that reduces $\mathcal{F}_{\mathrm{CC}}$ to the $\mathcal{F}_{\mathrm{EXTCOM}}$ functionality in a black-box way.

### 1.1.2 One-Sided UC Secure Computation

In this stronger two-party setting, where at most one of the parties is adaptively corrupted [KO04, HP14], we prove that one-sided adaptive UC security is implied by PKE with oblivious ciphertext generation. Here we combine two observations, one where our malicious static oblivious transfer from the previous result requires using the parties' inputs in only one phase, together with the fact that one-sided non-committing encryption (NCE) can be designed based on PKE with oblivious ciphertext generation [CFGN96, DN00]. In particular, NCE allow secure communication in the presence of adaptive attacks, which implies that the communication can be equivocated once the real message is handed to the simulator. Then, by encrypting part of our statically secure protocol using NCE, we obtain a generic protocol for any two-party functionality under the assumption specified above.[3] Namely,

[2] Where it suffices to realize the $\mathcal{F}_{\mathrm{OT}}$ functionality as it is known to be complete [Kil88].

[3] We note that while in the plain model any statically secure protocol can be compiled into one-sided secure protocol by encrypting its entire communication using one-sided NCE, it is not the case in the UC setting due to the additional setup.

**Theorem 1.3.** *(Informally) Assuming the existence of PKE with oblivious ciphertext generation, then any two-party functionality can be realized in the CRS model with one-sided adaptive UC security and black-box access to the PKE.*

### 1.1.3 Adaptive UC Secure Computation

Our last result is in the strongest corruption setting, where any number of parties can be adaptively corrupted. Here we design a new adaptively secure UC commitment scheme under the assumption of PKE with oblivious ciphertext generation, which is the first construction that achieves the stronger notion of adaptive security based on this hardness assumption. Our construction makes a novel usage of such a PKE together with Reed-Solomon codes, where the polynomial shares are encrypted using the PKE with oblivious ciphertext generation. Plugging-in our UC commitment protocol into the transformation of [CDMW09b] that generates adaptive malicious oblivious transfer given adaptive semi-honest oblivious transfer and UC commitments, implies an adaptively UC secure oblivious transfer protocol with malicious security based on semi-honest adaptive oblivious transfer and PKE with oblivious ciphertext generation, using only black-box access to the semi-honest oblivious transfer and the PKE. That is,

**Theorem 1.4.** *(Informally) Assuming the existence of PKE with oblivious ciphertext generation and adaptive semi-honest oblivious transfer, then any functionality can be realized in the CRS model with adaptive UC security, where the underlying primitives are accessed in a black-box manner.*

We further recall the work of Choi et al. [CDMW09a] that shows that the weakest general known assumption that is required to construct adaptively secure semi-honest oblivious transfer is trapdoor simulatable PKE. Now, since such an encryption scheme admits PKE with oblivious ciphertext generation, we obtain the following corollary that unifies the two assumptions required to achieve adaptive UC security.

**Corollary 1.5.** *Assuming the existence of (trapdoor) simulatable PKE, then any functionality can be realized in the CRS model with adaptive UC security and black-box access to the PKE.*

An additional interesting observation that is implied by our work is that our UC commitment scheme implies a construction that is secure in the adaptive setting when erasures are allowed, and under the weaker assumption of PKE. Specifically, instead of obliviously sampling ciphertexts in the commitment phase, the committer encrypts arbitrary plaintexts and then erases the plaintexts and randomness used for these computations. Our proof follows easily for this case as well. Combining our UC commitment scheme together with the semi-honest with erasures OT from [Lin09] and the transformation of [CDMW09b], we obtain the following result

**Theorem 1.6.** *(Informally) Assuming the existence of PKE and semi-honest oblivious transfer secure against an adaptive adversary assuming erasures, then any functionality can be realized in the CRS model with adaptive UC security assuming erasures, where the underlying primitives are accessed in a black-box manner.*

Noting that OT secure against adaptive adversaries assuming erasures can be realized under assumptions sufficient for achieving the same w.r.t the weaker static adversaries, this theorem shows that achieving UC-security against adaptive adversaries in the presence of erasures does not require any additional assumption beyond what is required to secure against static adversaries.

**Implications.** Next, we specify a sequence of interesting observations that are implied by our result in the adaptive UC setting.

- Previously, Dachman-Soled et al. [DMRV13], showed that adaptive UC secure protocols can be constructed in the CRS model assuming the existence of simulatable PKE. Our result improves this result in terms of complexity assumptions by showing that trapdoor simulatable PKE is sufficient, and provides new constructions based on concrete assumptions that were not known before. Nevertheless, we should point out that while the work of Dachman-Soled et al. is constructed in the global CRS model using a non-black-box construction, our result provides a black-box construction in a CRS model where the length of the reference string is linear in the number of parties.

- Analogous to our result on static UC security, it is possible to extend this result to the chosen key-registration authority (KRA) model, where we assume the existence of a trusted-party that samples public keys and secret keys for each party, and broadcasts the public key to all parties.

- It is important to note that this result provides the first evidence that adaptively secure UC commitment is theoretically easier to construct than stand-alone adaptively secure semi-honest oblivious transfer. This is due to a separation from [LZ09] (regarding static vs. adaptive oblivious transfer), that proves that adaptive oblivious transfer requires a stronger hardness assumption than enhanced trapdoor permutation.

- Regarding concrete assumptions, previously, adaptive UC commitments were constructed based on $N$-residuosity and $p$-subgroup hardness assumptions [DN02] and Strong RSA [DG03]. On the other hand, our result demonstrates the feasibility of this primitive under DDH, LWE, factoring Blum integers and RSA assumptions.

# 2 Preliminaries

**Basic notations.** We denote the security parameter by $n$. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large $n$ it holds that $\mu(n) < \frac{1}{p(n)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. We further denote by $a \leftarrow A$ the random sampling of $a$ from a distribution $A$, and by $[n]$ the set of elements $\{1, \ldots, n\}$. We specify next the definition of computationally indistinguishable.

**Definition 2.1.** *Let $X = \{X(a,n)\}_{a\in\{0,1\}^*, n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\{0,1\}^*, n\in\mathbb{N}}$ be two distribution ensembles. We say that $X$ and $Y$ are* computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if for every* PPT *machine $D$, every $a \in \{0,1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large $n$:*

$$\left|\Pr\left[D(X(a,n), 1^n) = 1\right] - \Pr\left[D(Y(a,n), 1^n) = 1\right]\right| < \frac{1}{p(n)}.$$

## 2.1 Public-Key Encryption Schemes

We specify the definitions of public-key encryption, IND-CPA and public-key encryption with oblivious ciphertext generation.

**Definition 2.2** (PKE)**.** *We say that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a* public-key encryption scheme *if* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are polynomial-time algorithms specified as follows:*

- $\mathsf{Gen}$*, given a security parameter $n$ (in unary), outputs keys* $(\mathrm{PK}, \mathrm{SK})$*, where* PK *is a public-key and* SK *is a secret key. We denote this by* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$.

- $\mathsf{Enc}$*, given the public-key* PK *and a plaintext message* $m$*, outputs a ciphertext* $c$ *encrypting* $m$*. We denote this by* $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m)$*; and when emphasizing the randomness* $r$ *used for encryption, we denote this by* $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; r)$*.*
- $\mathsf{Dec}$*, given the public-key* PK*, secret key* SK *and a ciphertext* $c$*, outputs a plaintext message* $m$ *s.t. there exists randomness* $r$ *for which* $c = \mathsf{Enc}_{\mathrm{PK}}(m; r)$ *(or* $\perp$ *if no such message exists). We denote this by* $m \leftarrow \mathsf{Dec}_{\mathrm{PK},\mathrm{SK}}(c)$*.*

For a public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a non-uniform adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the following *indistinguishability game*:

$(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$.
$(m_0, m_1, history) \leftarrow \mathcal{A}_1(\mathrm{PK})$, s.t. $|m_0| = |m_1|$.
$c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m_b)$, where $b \leftarrow_R \{0, 1\}$.
$b' \leftarrow \mathcal{A}_2(c, history)$.
$\mathcal{A}$ wins if $b' = b$.

Denote by $\mathrm{ADV}_{\Pi,\mathcal{A}}(n)$ the probability that $\mathcal{A}$ wins the IND-CPA game.

**Definition 2.3** (IND-CPA)**.** *A public-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-CPA secure*, if for every non-uniform adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $\mu(\cdot)$ *such that for all sufficiently large* $n$*'s,* $\mathrm{ADV}_{\Pi,\mathcal{A}}(n) \leq \frac{1}{2} + \mu(n)$*.*

A public-key encryption with the property of oblivious ciphertext generation implies additional two algorithms: (1) oblivious ciphertext generator $\widetilde{\mathsf{Enc}}$ and (2) a corresponding ciphertext faking algorithm $\widetilde{\mathsf{Enc}}^{-1}$. Intuitively, the ciphertext faking algorithm is used to explain a legitimately generated ciphertext as an obliviously generated one. Formally,

**Definition 2.4** (PKE with oblivious ciphertext generation [DN00])**.** *A PKE* $\Pi$ *with oblivious sampling generation is defined by the tuple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Enc}}^{-1})$ *and has the following additional property,*

- **Indistinguishability of oblivious and real ciphertexts.** *For any message* $m$ *in the appropriate domain, consider the experiment* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$*,* $c_1 \leftarrow \widetilde{\mathsf{Enc}}_{\mathrm{PK}}(r_1)$*,* $c_2 \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; r_2)$*,* $r_1' \leftarrow \widetilde{\mathsf{Enc}}^{-1}_{\mathrm{PK}}(c_2)$*. Then,* $(\mathrm{PK}, r_1', c_1, m) \overset{\mathrm{c}}{\approx} (\mathrm{PK}, r_2, c_2, m)$*.*

To this end, we only employ encryption schemes with perfect decryption. This merely simplifies the analysis and can be relaxed by using PKE with a negligible decryption error instead.

## 2.2 Secret-Sharing

A secret-sharing scheme allows distribution of a secret among a group of $n$ players, each of whom in a *sharing phase* receive a share (or piece) of the secret. In its simplest form, the goal of secret-sharing is to allow only subsets of players of size at least $t + 1$ to reconstruct the secret. More formally a $t + 1$-out-of-$n$ secret sharing scheme comes with a sharing algorithm that on input a secret $s$ outputs $n$ shares $s_1, \ldots, s_n$ and a reconstruction algorithm that takes as input $(s_i)_{i \in S}, S$ where $|S| > t$ and outputs either a secret $s'$ or $\perp$. In this work, we will use the Shamir's secret sharing scheme [Sha79] with secrets in $\mathbb{F} = GF(2^n)$. We present the sharing and reconstruction algorithms below:

**Sharing algorithm:** For any input $s \in \mathbb{F}$, pick a random polynomial $f(\cdot)$ of degree $t$ in the polynomial-field $\mathbb{F}[x]$ with the condition that $f(0) = s$ and output $f(1), \ldots, f(n)$.

**Reconstruction algorithm:** For any input $(s'_i)_{i\in S}$ where none of the $s'_i$ are $\perp$ and $|S| > t$, compute a polynomial $g(x)$ such that $g(i) = s'_i$ for every $i \in S$. This is possible using Lagrange interpolation where $g$ is given by

$$g(x) = \sum_{i\in S} s'_i \prod_{j\in S/\{i\}} \frac{x-j}{i-j} .$$

Finally the reconstruction algorithm outputs $g(0)$.

**Reed-Solomon code:** For integers $t, n$ and field $\mathbb{F}$, satisfying $0 < t \leq n < |\mathbb{F}|$, and a set of $n$ distinct elements $I = \{x_1, \ldots, x_n\} \subset \mathbb{F}$, the *Reed-Solomon code* $\mathcal{W}_{n,t}$ is defined by

$$\big\{q(x_1), \ldots, q(x_n) \mid q(\cdot) \text{ is a degree } t \text{ polynomial in } \mathbb{F}[x]\big\}.$$

The Reed-Solomon code has minimum distance relative distance $1-\frac{t}{n}$ where a corrupted codeword with up to $\lceil \frac{n-t}{2} \rceil$ errors can be corrected using the Berlekamp-Welch algorithm. It follows easily that Shamir's secret sharing on $\mathbb{F}$ as described above results in a sequence of shares in the Reed-Solomon code $\mathcal{W}_{n,t}$.

## 2.3 Oblivious Transfer

1-out-of-2 oblivious transfer (OT) is an important functionality in the context of secure computation that is engaged between a sender $\mathrm{Sen}$ and a receiver $\mathrm{Rec}$; see Figure 1 for the description of functionality $\mathcal{F}_{\mathrm{OT}}$. In this paper we are interested in reducing the hardness assumptions for general UC secure computation when using only black-box access to the underlying cryptographic primitives, such as the semi-honest OT. We use semi-honest OT as a building block for designing UC secure protocols in both static and adaptive settings. In the static setting, we refer to the two-round protocol of [EGL85] that is based on PKE with oblivious ciphertext generation (or enhanced trapdoor permutation). In the adaptive setting, we refer to the two-round protocol of [CLOS02] that is based on augmented non-committing encryption scheme.

We next recall that any two-round semi-honest OT implies PKE. We demonstrate that in two phases, starting with the claim that semi-honest OT implies a key agreement (KA) protocol, where two parties agree on a secret key over a public channel. This statement has already been proven in [GKM$^+$00] in the static setting, and holds for any number of rounds. The idea is simple, the parties execute an OT protocol where the party that plays the sender picks two random inputs $s_0, s_1$, whereas the party that plays the receiver enters 0. Finally, the parties output $s_0$ and security follows from the correctness and privacy of the OT. A simple observation shows that this reduction also holds in the adaptive setting. Namely, starting with an adaptive semi-honest OT, the same reduction implies an adaptively secure KA (where the protocol communication must be consistent with respect to any key). Note that this reduction preserves the number of rounds, thus if the starting point is a two-round OT then the reduction implies a two-round KA. Next, a well established fact shows that in the static setting a two-round key agreement implies PKE (in fact, these primitives are equivalent). Formally,

**Theorem 2.5.** *Assume the existence of two-round key agreement protocol with static security, then there exists IND-CPA PKE.*

### 2.3.1 Sender Private Oblivious Transfer

Sender privacy is a weaker notion than malicious security and only requires that the receiver's input be hidden even against a malicious sender. It is weaker than malicious security in that it does not require a simulation of the malicious sender that extracts the sender's inputs. In particular, we will only require that a malicious sender cannot distinguish the cases where the receiver's input is 0 or 1. Formally stated,

**Functionality $\mathcal{F}_{\text{OT}}$**

Functionality $\mathcal{F}_{\text{OT}}$ communicates with with sender Sen and receiver Rec, and adversary $\mathcal{S}$.

1. Upon receiving input $(\mathsf{sender}, sid, v_0, v_1)$ from Sen where $v_0, v_1 \in \{0,1\}^t$, record $(sid, v_0, v_1)$.
2. Upon receiving $(\mathsf{receiver}, sid, u)$ from Rec, where a tuple $(sid, v_0, v_1)$ is recorded and $u \in \{0,1\}$, send $(sid, v_u)$ to Rec and $sid$ to $\mathcal{S}$. Otherwise, abort.

Figure 1: The oblivious transfer functionality.

**Definition 2.6** (Sender private OT)**.** *Let $\pi$ be a two-party protocol that is engaged between a sender* Sen *and a receiver* Rec*. We say that $\pi$ is a* sender private *oblivious transfer protocol, if for every* PPT *adversary $\mathcal{A}$ that corrupts* Sen*, the following ensembles are computationally indistinguishable:*

- $\{\mathbf{View}_{\mathcal{A},\pi}[\mathcal{A}(1^n), \text{Rec}(1^n, 0)]\}_{n\in\mathbb{N}}$
- $\{\mathbf{View}_{\mathcal{A},\pi}[\mathcal{A}(1^n), \text{Rec}(1^n, 1)]\}_{n\in\mathbb{N}}$

*where* $\mathbf{View}_{\mathcal{A},\pi}[\mathcal{A}(1^n), \text{Rec}(1^n, b)]$ *denotes $\mathcal{A}$'s view within $\pi$ whenever the receiver* Rec *inputs the bit $b$.*

We point out that sender privacy protects the receiver against a malicious sender and should be read as privacy against a malicious sender.

### 2.3.2 Defensibly Private Oblivious Transfer

The notion of *defensible privacy* was introduced by Haitner in [Hai08, HIK$^+$11]. A defense in a two-party protocol $\pi = (P_1, P_2)$ execution is an input and random tape provided by the adversary after the execution concludes. A defense for a party controlled by the adversary is said to be *good*, if this party participated honestly in the protocol using this very input and random tape, then it would have resulted in the exact same messages that were sent by the adversary. In essence, this defense serves as a *proof* of honest behavior. It could very well be the case that an adversary deviates from the protocol in the execution but later provides a good defense. The notion of defensible privacy says that a protocol is private in the presence of defensible adversaries if the adversary learns nothing more than its prescribed output when it provides a good defense.

We begin with informally describing the notion of *good defense* for a protocol $\pi$; we refer to [HIK$^+$11] for the formal definition. Let $\mathsf{trans} = (q_1, a_1, \ldots, q_\ell, a_\ell)$ be the transcript of an execution of a protocol $\pi$ that is engaged between $P_1$ and $P_2$ and let $\mathcal{A}$ denote an adversary that controls $P_1$, where $q_i$ is the $i^{th}$ message from $P_1$ and $a_i$ is the $i^{th}$ message from $P_2$ (that is, $a_i$ is the response for $q_i$). Then we say that $(x, r)$ constitutes a *good defense* of $\mathcal{A}$ relative to $\mathsf{trans}$ if the transcript generated by running the honest algorithm for $P_1$ with input $x$ and random tape $r$ against $P_2$'s messages $a_1, \ldots, a_\ell$ results exactly in $\mathsf{trans}$.

The notion of defensible privacy can be defined for any secure computation protocol. Nevertheless, since we are only interested in oblivious transfer protocols, we present a definition below that is restricted to oblivious transfer protocols. The more general definition can be found in [HIK$^+$11]. At a high-level, an OT protocol is defensibly private with respect to a corrupted sender if no adversary interacting with an honest receiver with input $b$ should be able to learn $b$, if at the end of the execution the adversary produces any good defense. Similarly, an OT protocol that is defensibly private with respect to malicious receivers requires that any adversary interacting with an honest sender with input $(s_0, s_1)$ should not be able to learn $s_{1-b}$, if at the end of the execution the adversary produces a good defense with input $b$. Below we present a variant of the definition presented in [HIK$^+$11]. We stress that while the [HIK$^+$11] definition only considers bit OT (i.e. sender's inputs are bits) we consider *string OT*.

**Definition 2.7** (Defensible-private string OT)**.** *Let $\pi$ be a two-party protocol that is engaged between a sender* Sen *and a receiver* Rec*. We say that $\pi$ is a* defensibly-private *string oblivious transfer protocol, if for every* PPT *adversary $\mathcal{A}$ the following holds,*

1. $\{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathcal{A}(1^n), \mathrm{Rec}(1^n, U)], U)\} \overset{c}{\approx} \{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathcal{A}(1^n), \mathrm{Rec}(1^n, U)], U')\}$ *where $\Gamma(v, *)$ is set to $(v, *)$ if following the execution $\mathcal{A}$ outputs a good defense for $\pi$, and $\perp$ otherwise, and $U$ and $U'$ are independent random variables uniformly distributed over $\{0,1\}$. This property is referred to as defensibly private with respect to a corrupted sender.*

2. $\{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n, (U_0^n, U_1^n)), \mathcal{A}(1^n)], U_{1-b}^n)\} \overset{c}{\approx} \{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n, (U_0^n, U_1^n)), \mathcal{A}(1^n)], \bar{U}^n)\}$ *where $\Gamma(v, *)$ is set to $(v, *)$ if following the execution $\mathcal{A}$ outputs a good defense for $\pi$, and $\perp$ otherwise, $b$ is the* Rec*'s input in this defense and $U_0^n, U_1^n, \bar{U}^n$ are independent random variables uniformly distributed over $\{0,1\}^n$. This property is referred to as defensibly private with respect to a corrupted receiver.*

In our construction from Section 3, we will rely on an OT protocol that is sender private and defensibly private with respect to a corrupted receiver. In [HIK+11], Haitner et al. showed how to transform any semi-honest bit-OT to one that is defensibly private with respect to a corrupted receiver and malicious secure with respect to a corrupted sender. More formally, the following Lemma is implicit in the work of [HIK+11].

**Lemma 2.1** (Implicit in Theorem 4.1 and Corollary 5.3 [HIK+11])**.** *Assume the existence of a semi-honest oblivious transfer protocol $\pi$. Then there exists an oblivious transfer protocol $\hat{\pi}$ that is defensible-private with respect to the receiver and sender private that relies on the underlying primitive in a black-box manner.*

Now, since sender privacy is implied by malicious security with respect to a corrupted sender, this transformation yields a bit OT protocol with the required security guarantees. Nevertheless, our protocol crucially relies on the fact that the underlying OT is a string OT protocol. We therefore show in Appendix B how to transform any bit OT to a string OT protocol while preserving both defensible private with respect to a maliciously corrupted receiver and sender privacy.

At a high-level, in order to convert any protocol from semi-honest security to defensible privacy, Haitner et al. include a coin-tossing stage at the beginning of the protocol that determines the parties' random tapes. In fact, they let the coin-tossing also determine the parties inputs as they only require OT secure with respect to random inputs for both the sender and receiver. Now, if the receiver has to provide a good defense, then it must reveal the input and randomness used for the semi-honest OT protocol and prove consistency relative to the values generated in the coin-tossing stage. Due to the fact that the commitment schemes that are used in the coin-tossing stage are statistically-binding, the probability that a malicious receiver can deviate from the protocol and provide a good defense is negligible. Using this fact, Haitner et al. argued that the probability that a malicious receiver outputs a good defense and guesses the other sender's input is negligible. Next, to obtain sender private oblivious transfer they first transformed an OT protocol that is defensible-private against malicious receivers to one that is maliciously secure, and then exploited the symmetry of OT in order to obtain a protocol that is sender-private. The first transformation relies on the cut-and-choose approach to ensure that the receiver provides a valid defense, and then using the fact that defensible privacy hides the sender's other input they argued that it is receiver-private.

## 2.4 Commitment Schemes

Commitment schemes are used to enable a party, known as the *sender*, to commit itself to a value while keeping it secret from the *receiver* (this property is called hiding). Furthermore, in a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the

**Functionality $\mathcal{F}_{\mathrm{COM}}$**

Functionality $\mathcal{F}_{\mathrm{COM}}$ communicates with with sender Sen and receiver Rec, and adversary $\mathcal{S}$.

1. Upon receiving input $(\mathsf{commit}, sid, m)$ from Sen where $m \in \{0,1\}^t$, internally record $(sid, m)$ and send message $(sid, \mathrm{Sen}, \mathrm{Rec})$ to the adversary. Upon receiving approve from the adversary send $sid$, to Rec. Ignore subsequent $(\mathsf{commit}, ., ., .)$ messages.
2. Upon receiving $(\mathsf{reveal}, sid)$ from Sen, where a tuple $(sid, m)$ is recorded, send message $m$ to adversary $\mathcal{S}$ and Rec. Otherwise, ignore.

Figure 2: The string commitment functionality.

committing phase (this property is called binding). In this work, we consider commitment schemes that are statistically-binding, namely while the hiding property only holds against computationally bounded (non-uniform) adversaries, the binding property is required to hold against unbounded adversaries. More precisely, a pair of PPT machines Com is said to be a commitment scheme if the following two properties hold.

**Computational hiding:** For every (expected) PPT machine $R^*$, it holds that, the following ensembles are computationally indistinguishable over $n \in N$.

- $\{\mathsf{view}^{R^*}_{\mathsf{Com}}(v_1, z)\}_{n \in N, v_1, v_2 \in \{0,1\}^n, z \in \{0,1\}^*}$
- $\{\mathsf{view}^{R^*}_{\mathsf{Com}}(v_2, z)\}_{n \in N, v_1, v_2 \in \{0,1\}^n, z \in \{0,1\}^*}$

where $\mathsf{view}^{R^*}_{\mathsf{Com}}(v, z)$ denotes the random variable describing the output of $R^*$ after receiving a commitment to $v$ using Com.

**Statistical binding:** Informally, the statistical-binding property asserts that, with overwhelming probability over the coin-tosses of the receiver $R$, the transcript of the interaction fully determines the value committed to by the sender.

We say that a commitment is *valid* if there exists a unique committed value that a (potentially malicious) committer can open to successfully. We refer the reader to [Gol01] for more details.

## 2.5 UC Commitment Schemes

The notion of UC commitments was introduced by Canetti and Fischlin in [CF01]. The formal description of functionality $\mathcal{F}_{\mathrm{COM}}$ is depicted in Figure 2.

## 2.6 Extractable Commitments

Our result in the static setting requires the notion of (static) extractable UC commitments, which is a weaker security property than UC commitments in the sense that it does not require equivocality. Namely, the simulator is not required to commit to one message and then later convince the receiver that it committed to a different value. It is a real challenge to define this notion since it is hard to capture the notion of extractability in the ideal setting. In what follows, we introduce the definition for the ideal functionality $\mathcal{F}_{\mathrm{EXTCOM}}$ from [MPR10]. To the best of our knowledge, this is the only definition that captures straight-line extrcatability, statistically binding and computationally (stand-alone) hiding. Towards introducing this definition, Maji et al. introduced some notions first. More concretely,

**Definition 2.8.** *A protocol is a* syntactic commitment protocol *if:*

- *It is a two phase protocol between a sender and a receiver (using only plain communication channels).*
- *At the end of the first phase (commitment phase), the sender and the receiver output a transcript* $\mathsf{trans}$*. Furthermore, the sender receives an output (which will be used for opening the commitment).*
- *In the decommitment phase the sender sends a message* $\gamma$ *to the receiver, who extracts an output value* $\mathsf{opening}(\mathsf{trans}, \gamma) \in \{0,1\}^n \cup \{\perp\}$.

**Definition 2.9.** *Two syntactic commitment protocols* $(\omega_L, \omega_R)$ *form a pair of complementary statistically binding commitment protocols if the following hold:*

- $\omega_R$ *is a statistically binding commitment scheme (with stand-alone security).*
- *In* $\omega_L$*, at the end of the commitment phase the receiver outputs a string* $z \in \{0,1\}^n$*. If the receiver is honest, it is only with negligible probability that there exists* $\gamma$ *such that* $\mathsf{opening}(\mathsf{trans}, \gamma) \neq \perp$ *and* $\mathsf{opening}(\mathsf{trans}, \gamma) \neq z$.

As noted in [MPR10], $\omega_L$ by itself is not an interesting cryptographic goal, as the sender can simply send the committed string in the clear during the commitment phase. Nevertheless, in defining $\mathcal{F}_{\text{EXTCOM}}$ below, there exists a single protocol that satisfies both the security guarantees. We are now ready to introduce the notion of extractable commitments in Figure 3 that is parameterized by $(\omega_L, \omega_R)$. We additionally include a function $\mathsf{pp}$ that will be used as an initialization phase to set up the public-parameters for $\omega_L$ and $\omega_R$.

In Appendix C we show how to realize $\mathcal{F}_{\text{EXTCOM}}$ based on IND-CPA secure PKE.

# 3 Static UC Secure Computation

In this section we prove the feasibility of UC secure computation based on semi-honest OT and extractable commitments, where the latter can be constructed based on two-round semi-honest OT (see Sections 2.3 and 2.6 for more details). More concretely, we prove how to transform any statically semi-honest secure OT into one that is secure in the presence of malicious adversaries, giving *only black-box access* to the underlying semi-honest OT protocol. Our protocol is a variant of the protocol by Lin and Pass from [LP12] (which in turn is a variant of the protocol of [HIK$^+$11]). In particular, in [LP12], the authors rely on a strong variant of a commitment scheme known as a CCA-secure commitment in order to achieve extraction. We observe that it is not required to use the full power of such commitments, or for that matter UC commitments. Specifically, using a weaker primitive that only implies straight-line input extractability enables to solely rely on semi-honest OT. An important weakening in our commitment scheme compared to CCA-secure commitments from [LP12] is that we allow invalid commitments to be made by the adversary. We remark here that the work of [LP12] rely on string OT that are secure against malicious senders and state that the work of [Hai08] provides a black-box construction of such a protocol starting from a semi-honest bit OT. However, the work of [Hai08] only shows how to construct a bit OT secure against malicious senders where the proof crucially relies on the sender's input being only bits. We provide a transformation and complete analysis from bit OT to a string OT for the weaker notion of defensible privacy as this is sufficient for our work. Finally, combining our UC OT protocol with the [IPS08] protocol, we obtain a statically UC secure protocol for any well-formed functionality (see definition in [CLOS02]). Namely,

**Theorem 3.1.** *Assume the existence of static semi-honest oblivious transfer. Then for any multi-party well-formed functionality* $\mathcal{F}$*, there exists a protocol that UC realizes* $\mathcal{F}$ *in the presence of static, malicious adversaries in the* $\mathcal{F}_{\text{EXTCOM}}$*-hybrid model using black-box access to the oblivious transfer protocol.*

**Functionality $\mathcal{F}_{\text{EXTCOM}}$ parameterized by $(\mathsf{pp}, \omega_L, \omega_R)$**

$\mathcal{F}_{\text{EXTCOM}}$ is running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$: Upon receiving a message $(\mathsf{init}-\mathsf{commit}, sid, ssid, P_i, P_j)$ from $P_i$, it first checks if there is a tuple $(\mathsf{public}-\mathsf{params}, sid, P_i, (pp, sp))$. If yes, it sends $(\mathsf{init}-\mathsf{commit}, sid, ssid, P_i, P_j)$ to $P_j$. If not, it runs $(pp, sp) \leftarrow \mathsf{pp}(1^n)$ and sends $(\mathsf{init}-\mathsf{commit}, sid, P_i, pp)$ to $P_i$, $P_j$ and $\mathcal{S}$. It stores $(\mathsf{public}-\mathsf{params}, sid, P_i, (pp, sp))$. We denote $P_i$ by the sender and $P_j$ by the receiver in this interaction. Next, the functionality behaves as follows, depending on which party is corrupted.

- $P_i$ IS HONEST AND $P_j$ IS HONEST.

  **Commit Phase:** Upon receiving $(\mathsf{commit}, sid, ssid, P_i, P_j, m)$ from $P_i$, it internally simulates a session of $\omega_R$ (simulating both the sender and receiver in $\omega_R$), with the sender's input fixed to $m$. It gives $(\mathsf{transcript}, sid, ssid, \mathsf{trans}, \gamma)$ to $P_i$ and $(\mathsf{receipt}, sid, ssid, P_i, P_j, \mathsf{trans})$ to $P_j$ and $\mathcal{S}$.

  **Reveal Phase:** Upon receiving $(\mathsf{decommit}, sid, ssid, \cdot)$ from the sender, it sends $(\mathsf{decommit}, sid, ssid, P_i, P_j, z)$ to $P_j$ and $\mathcal{S}$.

- $P_i$ IS CORRUPTED AND $P_j$ IS HONEST.

  **Commit Phase:** It runs the commitment $\omega_L$ with the sender, playing the part of the receiver in $\omega_L$, to obtain $(sid, ssid, \mathsf{trans}, z)$. It sends $(\mathsf{receipt}, sid, ssid, P_i, P_j, \mathsf{trans})$ to $P_j$ and $\mathcal{S}$.

  **Reveal Phase:** Upon receiving $(\mathsf{decommit}, sid, ssid, \gamma)$ from the sender, if $\mathsf{opening}(\mathsf{trans}, \gamma) = z$, it sends $(\mathsf{decommit}, sid, ssid, P_i, P_j, z)$ to $P_j$ and $\mathcal{S}$. Otherwise ignore.

- $P_i$ IS HONEST AND $P_j$ IS CORRUPT.

  **Commit Phase:** Upon receiving $(\mathsf{commit}, sid, ssid, P_i, P_j, m)$ from $P_i$, it runs the commitment phase of $\omega_R$ with $P_j$, playing the sender's role in $\omega_R$ with $m$ as input. It obtains the output $(\mathsf{trans}, \gamma)$ at the end of this phase, and sends $(\mathsf{transcript}, sid, ssid, \mathsf{trans}, \gamma)$ to $P_i$.

  **Reveal Phase:** Upon receiving $(\mathsf{decommit}, sid, ssid)$ from the sender it sends $(\mathsf{decommit}, sid, ssid, P_i, P_j, (\gamma, z))$ to $P_j$ and $\mathcal{S}$.

The functionality does not do anything when both the sender and the receiver are corrupted.

Figure 3: Extractable commitment functionality.

We remark here that the work of [CDMW09b] shows how starting from a semi-honest oblivious transfer it is possible to obtain a black-box construction of an OT protocol that is secure against stand-alone static adversaries in the $\mathcal{F}_{\text{COM}}$-hybrid model. It is noted in [CDMW09b] that the (high-level) analysis provided in the work might be extendable to the UC-setting (cf. Footnote 10 in [CDMW09b]). Furthermore, in the static setting, it is conceivable that $\mathcal{F}_{\text{COM}}$ can be directly realized in the $\mathcal{F}_{\text{EXTCOM}}$-hybrid using the notion of extractable trapdoor commitments [PW09]. We do not pursue this approach and instead directly realize OT in the $\mathcal{F}_{\text{EXTCOM}}$-hybrid. While the previous works of [CDMW09b] and [HIK$^+$11] require a three step transformation, our transformation is one shot and therefore more direct.

It seems possible to generalize our theorem to multi-session functionalities. Analogous to [CF01], this will allows us to extend our corollaries to the Global CRS model by additionally assuming CCA encryption

scheme and leave it as future work.

### 3.1 Static UC Oblivious Transfer

In the following, we discuss a secure implementation of the oblivious transfer functionality (see Figure 1) with static, malicious security in the $\mathcal{F}_{\text{EXTCOM}}$-hybrid model (where $\mathcal{F}_{\text{EXTCOM}}$ is stated formally in Figure 3). Our goal in this section is to show that the security of malicious UC OT can be based on UC semi-honest OT, denoted by $\pi_{\text{OT}}^{\text{SH}}$, and extractable commitments. Our result is shown in two phases. At first we compile the semi-honest OT protocol $\pi_{\text{OT}}^{\text{SH}}$ into a new protocol with the security properties that are specified in Section 2.3.2, extending the [HIK$^+$11] transformation into string OT; denote the compiled OT protocol by $\widehat{\pi}_{\text{OT}}$. Next, we use $\widehat{\pi}_{\text{OT}}$ in order to construct a new protocol $\pi_{\text{OT}}^{\text{ML}}$ that is secure in the presence of malicious adversaries. Details follow,

**Protocol 1** (Protocol $\pi_{\text{OT}}^{\text{ML}}$ with static security)**.**

**Input:** *The sender* Sen *has input* $(v_0, v_1)$ *where* $v_0, v_1 \in \{0,1\}^n$ *and the receiver* Rec *has input* $u \in \{0,1\}$.

**The protocol:**

1. **Coin tossing:**

   - Receiver's random tape generation: *The parties use a coin tossing protocol in order to generate the inputs and random tapes for the receiver.*
     - *The receiver commits to* $20n$ *strings of appropriate length, denoted by* $a_{\text{Rec}}^1, \ldots, a_{\text{Rec}}^{20n}$*, by sending* $\mathcal{F}_{\text{EXTCOM}}$ *the message* $(\mathsf{commit}, sid, \widetilde{ssid_i}, a_{\text{Rec}}^i)$ *for all* $i \in [n]$.
     - *The sender responds with* $20n$ *random strings of appropriate length* $b_{\text{Rec}}^1, \ldots, b_{\text{Rec}}^{20n}$.
     - *The receiver computes* $r_{\text{Rec}}^i = a_{\text{Rec}}^i \oplus b_{\text{Rec}}^i$ *and then interprets* $r_{\text{Rec}}^i = c_i || \tau_{\text{Rec}}^i$ *where* $c_i$ *determines the receiver's input for the* $i^{th}$ *OT protocol, whereas* $\tau_{\text{Rec}}^i$ *determines the receiver's random tape used for this execution.*
   - Sender's random tape generation: *The parties use a coin tossing protocol in order to generate the inputs and random tapes for the sender.*
     - *The sender commits to* $20n$ *strings of appropriate length, denoted by* $a_{\text{Sen}}^1, \ldots, a_{\text{Sen}}^{20n}$*, by sending* $\mathcal{F}_{\text{EXTCOM}}$ *the message* $(\mathsf{commit}, sid, \widetilde{ssid'_i}, a_{\text{Sen}}^i)$ *for all* $i \in [n]$.
     - *The receiver responds with* $20n$ *random strings of appropriate length* $b_{\text{Sen}}^1, \ldots, b_{\text{Sen}}^{20n}$.
     - *The sender computes* $r_{\text{Sen}}^i = a_{\text{Sen}}^i \oplus b_{\text{Sen}}^i$ *and then interprets* $r_{\text{Sen}}^i = s_i^0 || s_i^1 || \tau_{\text{Sen}}^i$ *where* $(s_i^0, s_i^1)$ *determine the sender's input for the* $i^{th}$ *OT protocol, whereas* $\tau_{\text{Sen}}^i$ *determines the sender's random tape used for this execution.*

2. **Oblivious transfer:**

   - *The parties participate in* $20n$ *executions of the OT protocol* $\widehat{\pi}_{\text{OT}}$ *with the corresponding inputs and random tapes obtained from Stage 2. Let the output of the receiver in the* $i^{th}$ *execution be* $\tilde{s_i}$.

3. **Cut-and-choose:**

   - Sen *chooses a random subset* $q_{\text{Sen}} = (q_{\text{Sen}}^1, \ldots, q_{\text{Sen}}^n) \in \{1, \ldots, 20\}^n$ *and sends it to* Rec. *The string* $q_{\text{Sen}}$ *is used to define a set of indices* $\Gamma_{\text{Sen}} \subset \{1, \ldots, 20n\}$ *of size* $n$ *in the following way:* $\Gamma_{\text{Sen}} = \{20i - q_{\text{Sen}}^i\}_{i \in [n]}$. *The receiver then opens the commitments from Stage 1 that correspond to the indices within* $\Gamma_{\text{Sen}}$*, namely, the receiver decommits* $a_{\text{Rec}}^i$ *for all* $i \in \Gamma_{\text{Sen}}$. Sen *checks that the decommitted values are consistent with the inputs and randomness used for the OTs in Stage 2 by the receiver, and aborts in case of a mismatch.*
   - Rec *chooses a random subset* $q_{\text{Rec}} = (q_{\text{Rec}}^1, \ldots, q_{\text{Rec}}^n) \in \{1, \ldots, 20\}^n$ *and sends it to* Sen. *The string* $q_{\text{Rec}}$ *is used to define a set of indices* $\Gamma_{\text{Rec}} \subset \{1, \ldots, 20n\}$ *of size* $n$ *in the following way:* $\Gamma_{\text{Rec}} = \{20i - q_{\text{Rec}}^i\}_{i \in [n]}$. *The sender then opens the commitments from Stage 1 that correspond to the indices*

*within $\Gamma_{\mathrm{Rec}}$, namely, the sender decommits $a^i_{\mathrm{Sen}}$ for all $i \in \Gamma_{\mathrm{Rec}}$.* Rec *checks that the decommitted values are consistent with the inputs and randomness used for the OTs in Stage 2 by the sender, and aborts in case of a mismatch.*

- Rec *commits to another subset $\Gamma \subset [20n]$ denoted by $(\Gamma^1, \ldots, \Gamma^n)$, by sending $\mathcal{F}_{\mathrm{EXTCOM}}$ the message $(\mathsf{commit}, sid, ssid'_i, \Gamma^i)$ for all $i \in [n]$. (The sender will reveal its inputs and randomness that are used in Stage 2 that correspond to the indices in $\Gamma$ later in Stage 5.)*

4. **Combiner:**

   - *Let $\Delta = [20n] - \Gamma_{\mathrm{Rec}} - \Gamma_{\mathrm{Sen}}$. Then for every $i \in \Delta$, the receiver computes $\alpha_i = u \oplus c_i$ and sends it to the sender.*
   - *The sender computes a $10n$-out-of-$18n$ secret sharing of $v_0$, denote the shares by $\{\rho^0_i\}_{i\in\Delta}$. Analogously, it computes a $10n$-out-of-$18n$ secret sharing of $v_1$, denote the shares by $\{\rho^1_i\}_{i\in\Delta}$. The sender computes $\beta^b_i = \rho^b_i \oplus s_i^{b\oplus\alpha_i}$ for all $b \in \{0,1\}$ and $i \in \Delta$, and sends the outcome to the receiver.*
   - *The receiver computes $\tilde{\rho}_i = \beta^u_i \oplus \tilde{s}_i$ for all $i \in \Delta$. Denote by $\rho$ these concatenated bits.*

5. **Final cut-and-choose:**

   - *The receiver decommits $\Gamma$ and the sender sends the inputs and randomness it used in Stage 2 for the coordinates that correspond to $\Delta \cap \Gamma$. (Note that the sender need only reveal the indices that were not decommitted in Stage 3).* Rec *checks that the sender's values are consistent with the inputs and randomness used for the OTs in Stage 2 by the sender, and aborts in case of a mismatch.*
   - *The receiver checks whether $(\tilde{\rho}_i)_{i\in\Delta}$ agrees with some codeword $w \in \mathcal{W}_{18n,10n}$ on $17n$ locations (where the code $\mathcal{W}_{18n,10n}$ is induced by the secret sharing construction that we use in Stage 4; see Definition 2.2 for more details). Recall that the minimum distance of the code $\mathcal{W}_{18n,10n}$ is at least $18n - 10n > 8n$, which implies that there will be at most one such codeword $w$. Furthermore, since we can correct up to $\frac{18n-10n}{2} = 4n$ errors, any code that is $17n$ close to a codeword can be efficiently recovered using the Berlekamp-Welch algorithm. The receiver outputs that $w$ as its output in the OT protocol. If no such $w$ exists, the receiver returns a default value.*

We next prove the following theorem.

**Theorem 3.2.** *Assume that $\pi^{\mathrm{SH}}_{\mathrm{OT}}$ is static semi-honest secure and that the compiled $\widehat{\pi}_{\mathrm{OT}}$ is secure according to Lemma 2.1. Then Protocol 1 UC realizes $\mathcal{F}_{\mathrm{OT}}$ in the presence of static malicious adversaries in the $\mathcal{F}_{\mathrm{EXTCOM}}$-hybrid model using black-box access to the oblivious transfer protocol.*

Recalling that our protocol relies on the existence of semi-honest OT and extractable commitments, and that the later can be constructed based on any two-round semi-honest OT, e.g., [EGL85], which implies PKE (see Sections 2.3 and 2.6 for more details), an immediate corollary from Theorem 3.2 implies that,

**Corollary 3.3.** *Assume the existence of two-round static semi-honest oblivious transfer. Then there exists a protocol that securely realizes $\mathcal{F}_{\mathrm{OT}}$ in the presence of static malicious adversaries in the CRS model using black-box access to the oblivious transfer protocol.*

**A high level proof.** We first provide an overview of the security proof; the complete proof is found in Section 3.2. Loosely speaking, in case the receiver is corrupted the simulator plays the role of the honest sender in Stages 1-4. Next in Stage 5, the simulator extracts the receiver's input $u$. Specifically, the simulator extracts all the committed values of the receiver within Stage 1 (relying on the fact that the commitment scheme is extractable), and then uses these values in order to obtain the inputs for the OT executions in Stage 2. Upon completing Stage 2, the simulator records the coordinates for which the receiver deviates from the prescribed input and random tape chosen in the coin tossing phase. Denoting these set of coordinates by $\Phi$, we recall that a malicious receiver may obtain both of the sender's inputs with respect to the OT executions that correspond to the coordinates within $\Phi$ and $\Gamma$. On the other hand, it obtains only one of the

two inputs with respect to the rest of the OT executions that correspond to the coordinates within $\Delta - \Phi - \Gamma$. Consequently, the simulator checks how many shares of $v_0$ and $v_1$ are obtained by the receiver and proceeds accordingly. In more details,

- If the receiver obtains more than $10n$ shares of both inputs then the simulator halts and outputs fail (we prove in Section 3.2 that this event only occurs with negligible probability).
- If the receiver obtains less than $10n$ shares of both inputs then the simulator picks two random values for $v_0$ and $v_1$ of the appropriate length and completes the interaction, playing the role of the honest sender on these values. Note that in this case the simulator does not need to call the ideal functionality.
- Finally, if the receiver obtains more than $10n$ shares for only one input $u \in \{0, 1\}$, then the simulator sends $u$ to the ideal functionality $\mathcal{F}_{\text{OT}}$ and obtains $v_u$. The simulator then sets $v_{1-u}$ as a random string of the appropriate length and completes the interaction by playing the role of the honest sender on these values.

Recall that the only difference between the simulation and the real execution is in the way the messages in Stage 4 are generated. Specifically, in the simulation a value $u$ is extracted from the malicious receiver and then fed to the $\mathcal{F}_{\text{OT}}$ functionality. The simulation is then completed based on the output returned from the functionality. Intuitively, the cut-and-choose mechanism ensures that the receiver cannot deviate from the honest strategy in Stage 2 in more than $n$ OT sessions without getting caught with overwhelming probability. Moreover, the defensible privacy of the OT protocol implies that the receiver can learn at most one of the two inputs of the sender relative to the OT executions in Stage 2 for which the receiver proceeded honestly.

In case the sender is corrupted, the simulator's strategy is to play the role of the honest receiver until Stage 5 where the simulator extracts the sender's inputs. More specifically, the simulator first extracts the sender's input for the OT executions in Stage 1 (relying on the fact that the commitment scheme is extractable). Next, the simulator extracts the shares $\{\rho_i^0\}_{i\in\Delta}$ and $\{\rho_i^1\}_{i\in\Delta}$ that correspond to inputs $v_0$ and $v_1$. To obtain the actual values the simulator checks if these shares agree with some codeword relative to $16n$ locations. That is,

- Let $w_0$ and $w_1$ denote the corresponding codewords (if there are no such codewords that agree with with $v_0$ and $v_1$ on $16n$ locations then the simulator uses a default codeword instead). Next, the simulator checks $w_0$ and $w_1$ against the final cut-and-choose. If any of the shares from $w_b$ are inconsistent with the opened shares that are opened by the sender in the final cut-and-choose, then $v_b$ is set to a default value, otherwise $v_b$ is the value corresponding to the shared secret.

Finally, the simulator sends $(v_0, v_1)$ to the ideal functionality for $\mathcal{F}_{\text{OT}}$. Security in this case is reduced to the privacy of the receiver. In addition, the difference between the simulation's strategy and the honest receiver's strategy is that the simulator extracts the sender's both inputs in all $i \in \Delta - \Phi$ and then finds codewords that are $16n$-close to the extracted values, whereas the honest receiver finds a codeword that is $17n$-close based on the inputs it received in the Stages 2 and 5, and returns it. We thus prove that the value $u$ extracted by the simulator is identical the to the reconstructed output of the honest receiver relying on the properties of the secret sharing scheme.

## 3.2 Proof of Theorem 3.2

Let $\mathcal{A}$ be a malicious probabilistic polynomial-time real adversary running protocol 1 in the $\mathcal{F}_{\text{EXTCOM}}$-hybrid model. We construct an ideal model adversary $\mathcal{S}$ with access to $\mathcal{F}_{\text{OT}}$ which simulates a real execution of protocol $\pi_{\text{OT}}^{\text{ML}}$ with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal process with $\mathcal{S}$ and $\mathcal{F}_{\text{OT}}$ from a real execution of $\pi_{\text{OT}}^{\text{ML}}$ with $\mathcal{A}$. $\mathcal{S}$ start by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$

with environment $\mathcal{Z}$, emulating the honest party. We separately describe the actions of $\mathcal{S}$ actions for every corruption case.

**Simulating the communication with $\mathcal{Z}$:** Every message that $\mathcal{S}$ receives from $\mathcal{Z}$ it internally feeds to $\mathcal{A}$ and every output written by $\mathcal{A}$ is relayed back to $\mathcal{Z}$.

**Simulating the corrupted receiver:** In this case $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ emulates functionality $\mathcal{F}_{\text{EXTCOM}}$ in Stage 1 and invokes $20n$ times the commitment scheme $\omega_L$ with $\mathcal{A}$ (that plays the role of the committer), obtaining $((\widetilde{\mathsf{trans}}_1, a^1_{\text{Rec}}), \ldots, (\widetilde{\mathsf{trans}}_{20n}, a^{20n}_{\text{Rec}}))$. It internally records $a^1_{\text{Rec}}, \ldots, a^{20n}_{\text{Rec}}$ and further picks $20n$ random strings $b^1_{\text{Rec}}, \ldots, b^{20n}_{\text{Rec}}$, forwarding them to the adversary. The simulator also computes $r^i_{\text{Rec}} = a^i_{\text{Rec}} \oplus b^i_{\text{Rec}}$ and then views $r^i_{\text{Rec}} = c_i || \tau^i_{\text{Rec}}$ where $c_i$ is the input an honest receiver must use in the $i^{th}$ OT protocol execution in Stage 2, together with randomness $\tau^i_{\text{Rec}}$.

   Next, the simulator picks $20n$ random strings $a^1_{\text{Sen}}, \ldots, a^{20n}_{\text{Sen}}$ and emulates the ideal functionality $\mathcal{F}_{\text{EXTCOM}}$ by invoking $20n$ times the commitment phase of $\omega_R$ with inputs $a^1_{\text{Sen}}, \ldots, a^{20n}_{\text{Sen}}$, against $\mathcal{A}$ that plays the role of receiver for the commitment scheme. At the end of this phase $\mathcal{S}$ obtains the output $((\widetilde{\mathsf{trans}'_1}, \gamma_1), \ldots, (\widetilde{\mathsf{trans}'_{20n}}, \gamma_{20n}))$ and receives from the adversary $20n$ random strings $b^1_{\text{Sen}}, \ldots, b^{20n}_{\text{Sen}}$.

2. In Stage 2 the simulator participates with the adversary in $20n$ executions of the OT protocol $\widehat{\pi}_{\text{OT}}$, while playing the role of the honest sender. Note that due to the fact that the simulator knows the values of the input and randomness that the honest receiver must use in each of the OT executions, the simulator can identify the coordinates of which the receiver deviates. We denote this set of coordinates by the set $\Phi$.

3. In Stage 3 the simulator picks $n$ random numbers $(q^1_{\text{Sen}}, \ldots, q^n_{\text{Sen}})$ from $\{1, \ldots, 20\}^n$ and sends them to the receiver. Upon receiving the decommitments from the receiver, the simulator verifies the decommitments as would the honest sender do with respect to $\Gamma_{\text{Sen}}$ and aborts in case of a mismatch. Next, it receives $(q^1_{\text{Rec}}, \ldots, q^n_{\text{Rec}})$ from the receiver and decommits the subset of values that corresponds to the coordinates in $\Gamma_{\text{Rec}}$ as determined by $(q^1_{\text{Rec}}, \ldots, q^n_{\text{Rec}})$, playing the role of the sender. Finally, it emulates functionality $\mathcal{F}_{\text{EXTCOM}}$ and invokes the commitment scheme $\omega_L$ with $\mathcal{A}$ (that plays the role of the committer) $n$ times, obtaining $((\mathsf{trans}'_1, \Gamma_1), \ldots, (\mathsf{trans}'_{20n}, \Gamma_{20n}))$. Let $\Delta = [20n] - \Gamma_{\text{Rec}} - \Gamma_{\text{Sen}}$.

4. In Stage 4 the simulator proceeds as follows. Observe first that $\Phi$ and $\Gamma_{\text{Sen}}$ are disjoint, since otherwise the simulator would have aborted in the previous stage. We consider three cases here:

   (a) $\Phi \geq n$: In this case the simulator halts and outputs fail.

   (b) $\Phi < n$: This implies that $\Delta - \Phi - \Gamma > 16n$ where by definition, the malicious receiver proceeds according to the honest prover's strategy with respect to every coordinate in $\Delta - \Phi - \Gamma$. Note that in this case the adversary learns at most $\Delta + 2\Phi + 2\Gamma < 20n$ distinct shares of both the sender's inputs and the simulator knows precisely which share is learned for every coordinate relative to the set $\Delta - \Phi - \Gamma$. We consider two subcases:

      i. There exists a bit $u \in \{0, 1\}$ for which the adversary learns $10n$ shares of $v_u$. We recall that the adversary might learn both shares for the OT executions that correspond to the coordinates within the set $\Phi \cup \Gamma$, and exactly one share for every OT execution that corresponds to the coordinates within $\Delta - \Phi - \Gamma$. Now, since the simulator knows which share is learned for every coordinate within $\Delta - \Phi - \Gamma$, it can also compute an upper bound on the number of $v_u$ shares that are obtained by the receiver. In this case the simulator forwards $u$ to the ideal functionality $\mathcal{F}_{\text{OT}}$ and receives back $v_u$. The simulator then sets $v_{1-u}$ as a random string of

the appropriate length and completes the interaction by playing the role of the honest sender on these inputs.

ii. There does not exist a bit $u \in \{0,1\}$ for which the adversary learns $10n$ shares of $v_u$. In this case the simulator picks two random values for $v_0$ and $v_1$ of the appropriate length and completes the interaction, playing the role of the honest sender on these values. Note that in this case the simulator does not call the ideal functionality $\mathcal{F}_{\text{OT}}$.

5. The simulator completes the simulation in Stage 5 similarly to Stage 3.

Note that the only difference between the simulation and the real execution is in the way the messages in Stage 4 are generated. In the simulation, a value $u$ is extracted from the malicious receiver and then fed to the $\mathcal{F}_{\text{OT}}$ functionality, where the simulation is completed based on the output returned from the functionality. Furthermore, we recall that in Stage 5 the receiver learns both of the sender's inputs in all sessions $i \in \Gamma$, then it holds that the receiver learns one such input for every session it behaved honestly and two inputs for all sessions it deviates or included in $\Gamma$. Proving that the adversary deviates in more than $n$ OT executions only happens with negligible probability, implies that it learns less than $20n$ shares in total. Therefore, at least one of the shared secrets is completely hidden due to the $10n$-out-of-$18n$ secret sharing scheme. To complete the simulation, the simulator identifies which of the two values $v_0$ and $v_1$ is learnt by the receiver (by identifying how many shares are obtained by that party) and fixes that to be the receiver's input. Finally, indistinguishability follows from the defensible privacy with respect to the receiver of the OT protocol.

More formally, we begin with a proof that the probability that the simulator returns fail is negligible and then neglect this event. Namely, we prove that the simulated and real views are computationally indistinguishable conditioned on the even that the simulator did not output fail.

**Claim 3.1.** *The probability $\mathcal{S}$ return* fail *is negligible.*

**Proof:** Note first that the only place where the simulator fails is in Step 4a, when $\phi \geq n$. We now show that this event occurs with negligible probability. In other words, we need to show that the probability that the corrupted receiver deviates from the honest receiver's strategy in at least $n$-out-of-$18n$ OT executions while not getting caught by the sender is negligible. Formally, let $\text{Bad}$ denote the event for which the corrupted receiver deviates in at least $n$ coordinates. Note first that the simulator can easily identify when event $\text{Bad}$ occurs since it knows the random tapes and the inputs the receiver must use in all executions, and can therefore identify the coordinates for which the receiver deviates. Next, we show that conditioned on event $\text{Bad}$ occurring, the probability that $\Gamma_{\text{Sen}}$ does not contain one of the $n$ deviated coordinates is negligible. This implies that the probability that $\mathcal{S}$ returns fail is negligible.

Denote by $\Phi$ the set of $n$ coordinates in which the receiver deviates and define the bins $A_j = \{20(j-1)+1, \ldots, 20j\}$ for all $j \in [n]$. By the pigeon-hole principle it holds that at least $\lfloor n/20 \rfloor$ bins intersect with $\Phi$. In addition, we recall that $\Gamma_{\text{Sen}}$ is chosen by the simulator by picking one element from each bin independently of $\Phi$ and uniformly at random. Then the probability that $\Gamma_{\text{Sen}} \cap \Phi = \emptyset$ is at most $(19/20)^{\lfloor n/20 \rfloor}$ which is negligible in $n$. This concludes the proof of the Claim. □

Next, we prove that the receiver's view for both executions is computationally indistinguishable assuming that the simulator did not abort the execution. More formally, denoting the simulated execution by $\pi_{\text{IDEAL}}$, then we prove the following statement,

**Claim 3.2.** *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{F}_{\text{EXTCOM}}}_{\pi^{\text{ML}}_{\text{OT}},\mathcal{A},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\text{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\text{OT}}}_{\pi_{\text{IDEAL}},\mathcal{S},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** The security argument proceeds in a sequence of hybrids games starting from the simulated execution towards the real execution. We denote by $\pi_{\text{HYBRID}_i}$ the receiver's view in the $i^{th}$ hybrid game.

**Hybrid 1:** In this game we define a simulator $\mathcal{S}_1$ that is identical to $\mathcal{S}$ except for the way the sender's message is generated in Stage 4. More precisely, the simulator modifies the way $\beta_i^{1\oplus u}$ is computed for all $i \in \Delta - \Gamma - \Phi$. Recall first that $\mathcal{S}$ sets

$$\beta_i^{1\oplus u} = \rho_i^{1\oplus u} \oplus s_i^{(1\oplus u)\oplus \alpha_i} = \rho_i^{1\oplus u} \oplus s_i^{1\oplus c_i}.$$

Instead, $\mathcal{S}_1$ will choose $\beta_i^{1\oplus u}$ at random, which can be viewed as using a making element that is independent of $s_i^{1\oplus c_i}$. Intuitively, we claim that the simulated view and the view generated in hybrid 1 are computationally indistinguishable because for every $i \in \Delta - \Gamma - \Phi$ the receiver generates the OT messages in session $i$ of Stage 2 honestly (i.e. using the input $c_i$ and random tape $\tau^i_{\mathrm{Rec}}$), where by the defensible privacy of the receiver it cannot distinguish the input $s_i^{1\oplus c_i}$ from a random input. We formally prove this statement in the following claim.

**Claim 3.3.** *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{OT}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \equiv \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{OT}}}_{\pi_{\mathrm{HYBRID}_1},\mathcal{S}_1,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** Towards proving this claim, we introduce a sequence of intermediate hybrid experiments $H_1^e$ for $e = 0, \ldots, 18n$, where in hybrid $H_1^e$ we consider a simulator $\mathcal{S}_1^e$ that proceeds identically to $\mathcal{S}$ with the exception that it follows $\mathcal{S}_1$'s strategy for the first $e$ indices in $\Delta - \Gamma - \Phi$ regarding the generation of $\beta_i^{1\oplus u}$ (i.e. for the first $e$ sessions where the receiver proceeded honestly). By definition we have that experiment $H_1^0$ proceeds identically to the ideal simulation and $H_1^{18n}$ proceeds identically to hybrid 1. Denote the view output in hybrid $H_1^e$ by $\mathsf{hyb}_e(n)$ and assume by contradiction that there exist an adversary $\mathcal{A}$ (controlled by $\mathcal{Z}$), a distinguisher $D$, a polynomial $p(\cdot)$ and infinitely many $n$'s such that,

$$\left|\Pr[D(\mathbf{View}^{\mathcal{F}_{\mathrm{OT}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S},\mathcal{Z}}(n)) = 1] - \Pr[D(\mathbf{View}^{\mathcal{F}_{\mathrm{OT}}}_{\pi_{\mathrm{HYBRID}_1},\mathcal{S},\mathcal{Z}}(n)) = 1]\right| \geq \frac{1}{p(n)}\,.$$

Using a standard hybrid argument it follows that there exists an $e \in [18n]$ such that,

$$\left|\Pr[D(\mathsf{hyb}_e(n)) = 1] - \Pr[D(\mathsf{hyb}_{e-1}(n)) = 1]\right| \geq \frac{1}{18np(n)}\,.$$

Next, we plan to exploit the above observation in order to construct a defensible adversary $\mathcal{A}'$ that violates the receiver's defensible privacy relative to $\widehat{\pi}_{\mathrm{OT}}$ in the sense of Definition 2.7. At a high-level $\mathcal{A}'$ picks a random $j \in [20n]$ and externally forwards $\mathcal{A}$'s messages within the $j^{th}$ execution of the OT protocol, where $j$ serves as the guess for the $e^{th}$ execution in $\Delta - \Gamma - \Phi$. $\mathcal{A}'$ then emulates the rest of the OT executions, playing the role of the sender. In order to simplify the analysis and allow $\mathcal{A}'$ carry out the reduction properly (where the generated randomness within the coin-tossing phase is disassociated from the OT executions) we consider the following additional hybrid executions.

First, we consider a slight variation of $H_i^{e-1}$ (resp., $H_i^e$) denoted by $\overline{H}_{e-1}$ (resp., $\overline{H}_e$), and a random variable $J$ that denotes a randomly chosen index from $[3n+1]$ which is picked at the onset of the hybrid execution. Moreover, the experiment is aborted if chosen index does not correspond to the $e^{th}$ execution in $\Delta - \Gamma - \Phi$. We say that index $J$ is $\mathrm{Bad}$ if the experiment aborts. Note that experiments $\overline{H}_{e-1}$ and $\overline{H}_e$ proceed identically to $H_{e-1}$ and $H_e$, respectively, conditioned on $J$ not being $\mathrm{Bad}$. This is due to the fact that $J$ is chosen independently of the experiments. Moreover, relying on the fact that the $e^{th}$ execution can take at most $20n$ values we have that,

$$\Pr[J \text{ is not } \mathrm{Bad}] = \frac{1}{20\mathrm{n}}.$$

Therefore, if $\overline{\mathsf{hyb}}_{e-1}(n)$ and $\overline{\mathsf{hyb}}_{e}(n)$ respectively correspond to $\mathcal{A}$'s views in $\overline{H}_{e-1}$ and $\overline{H}_{e}$, then

$$\begin{aligned}\Pr[D(\overline{\mathsf{hyb}}_e(n))=1] &= \Pr[D(\overline{\mathsf{hyb}}_e(n))=1 \wedge \ J \text{ not Bad}] + \Pr[\mathrm{D}(\overline{\mathsf{hyb}}_{\mathrm{e}}(\mathrm{n}))=1 \wedge \ \mathrm{J} \text{ is Bad}] \\ &= \left(\frac{1}{20n}\right)\Pr[D(\overline{\mathsf{hyb}}_e(n))=1 \mid J \text{ not Bad}] + \left(1-\frac{1}{20\mathrm{n}}\right)\Pr[\mathrm{D}(\overline{\mathsf{hyb}}_{\mathrm{e}}(\mathrm{n}))=1 \mid \mathrm{J} \text{ is Bad}] \\ &= \left(\frac{1}{20n}\right)\Pr[D(\mathsf{hyb}_e(n))=1] + \left(1-\frac{1}{20n}\right)\Pr[D(\perp)=1].\end{aligned}$$

Similarly,

$$\Pr[D(\overline{\mathsf{hyb}}_{e-1}(n))=1] = \left(\frac{1}{20n}\right)\Pr[D(\mathsf{hyb}_{e-1}(n))=1] + \left(1-\frac{1}{20n}\right)\Pr[D(\perp)=1].$$

Therefore,

$$\begin{aligned}&\left|\Pr[D(\overline{\mathsf{hyb}}_e(n))=1] - \Pr[D(\overline{\mathsf{hyb}}_{e-1}(n))=1]\right| \\ &\qquad = \left(\frac{1}{20n}\right)\left|\Pr[D(\mathsf{hyb}_e(n))=1] - \Pr[D(\mathsf{hyb}_{e-1}(n))=1]\right| \\ &\qquad \geq \left(\frac{1}{20n}\right)\frac{1}{18np(n)}. \end{aligned} \tag{1}$$

Before we provide the description of $\mathcal{A}'$, we consider our second modification and define hybrids $\widetilde{H}_{e-1}$ and $\widetilde{H}_e$ as follows. Namely, in these new experiments we slightly modify the sender's messages in the coin-tossing phase and ask the sender to commit to the all zeros string of appropriate length instead of committing to a uniform string $a^j_{\mathrm{Sen}}$. Recalling that $a^j_{\mathrm{Sen}}$ and $b^j_{\mathrm{Sen}}$ determine the sender's input in the $j^{th}$ execution of the OT protocol, we instruct the sender to commit to 0 so that $\mathcal{A}'$ can forward the $j^{th}$'s execution messages to an external OT sender in the reduction described next. More precisely, $\widetilde{H}_{e-1}$ (resp., $\widetilde{H}_e$) follows exactly as $\overline{H}_{e-1}$ (resp., $\overline{H}_e$) with the exception that we modify the honest sender's message in the coin-tossing stage, where it commits to the all zeros string instead of $a^J_{\mathrm{Sen}}$. Observe that this change does not affect the cut-and-choose phase where the sender is required to reveal randomness for indices in $\Gamma_{\mathrm{Rec}}$ because if $J \in \Gamma_{\mathrm{Rec}} \wedge \Gamma$ then the experiment is aborted by definition. Denote by the random variables $\widetilde{\mathsf{hyb}}_e(n)$ and $\widetilde{\mathsf{hyb}}_{e-1}(n)$ the views of adversary $\mathcal{A}$ in $\widetilde{H}_{e-1}$ and $\widetilde{H}_e$, respectively. Then from the computational hiding property of $\omega_L$ the commitment scheme used in the coin-tossing stage it follows that[4] This is because to reduce the indistinguishability of there exists a negligible function $\nu(\cdot)$ such that for all sufficiently large $n$'s,

$$\left|\Pr[D(\overline{\mathsf{hyb}}_e(n))=1] - \Pr[D(\widetilde{\mathsf{hyb}}_e(n))=1]\right| \ \leq \ \nu(n) \tag{2}$$

$$\left|\Pr[D(\overline{\mathsf{hyb}}_{e-1}(n))=1] - \Pr[D(\widetilde{\mathsf{hyb}}_{e-1}(n))=1]\right| \ \leq \ \nu(n)\,. \tag{3}$$

Using Equation 1 we obtain that for all sufficiently large $n$'s,

$$\left|\Pr[D(\widetilde{\mathsf{hyb}}_e(n))=1] - \Pr[D(\widetilde{\mathsf{hyb}}_{e-1}(n))=1]\right| \geq \frac{1}{(18n)20np(n)} - 2\nu(n) \geq \frac{1}{q(n)} \tag{4}$$

where $q(\cdot)$ is the polynomial $2 \times 18(20n)np(n)$. Fix an $n$ such that this happens. We now show how to define $\mathcal{A}'$ and distinguisher $D'$ that violate the defensible private with respect to a corrupted

[4]We remark here that we rely on the security of the commitment scheme $\omega_L$ against receivers with auxiliary input as in Section 2.4.

receiver in $\pi_{\mathrm{OT}}^{\mathrm{ML}}$. More specifically, $\mathcal{A}'$ internally emulates experiment $\widetilde{H}_1^{e-1}$ by running the simulation strategy of $\mathcal{S}_1^{e-1}$ with the malicious receiver $\mathcal{A}$. Let $(c_J, \tau_{\mathrm{Rec}}^J)$ denote the input and randomness that the honest receiver is supposed to use in the internal $J^{th}$ execution. Recall that this is determined by $a_{\mathrm{Rec}}^J \oplus b_{\mathrm{Rec}}^J$ and is known to the simulator, as it extracts the adversary's commitments. Next $\mathcal{A}'$ plays the role of the sender in the executions of $\widehat{\pi}_{\mathrm{OT}}$ with the exception that it externally relays the messages of the adversary (acting as the receiver) in the $J^{th}$ execution of the oblivious transfer protocol from Stage 2. Following the oblivious transfer executions, $\mathcal{A}'$ continues the internal emulation until the end of Stage 3. If the experiment aborts in the internal emulation (where this happens if $J$ is $\mathrm{Bad}$) then $\mathcal{A}'$ aborts. Otherwise, there is a good defense for the receiver in the $J^{th}$ execution, namely $(c_J, \tau_{\mathrm{Rec}}^J)$. Let $\mathsf{STATE}$ be the complete view of experiment $\widetilde{H}_{e-1}$ which includes the input and random tape of $\mathcal{A}$ and the simulator (playing the sender), as well as the partial transcript of the messages exchanged with $\mathcal{A}$ until Stage 3. $\mathcal{A}'$ outputs $(c_J, \tau_{\mathrm{Rec}}^J)$ as its defense and $\mathsf{STATE}$ as its output.

Upon receiving $(\mathsf{view}, s)$, where view is $\mathcal{A}'$'s view and $s$ is a string (as specified in Definition 2.7), distinguisher $D'$ proceeds as follows. It first extracts state $\mathsf{STATE}$ from the view and then completes the internal emulation of the experiment by playing the role of the sender in Stages 4 and 5. We note that $D'$ has all the information it needs as part of $\mathsf{STATE}$ to complete the execution, except for the sender's inputs $(s_J^0, s_J^1)$ that are required to compute $\beta_J^0$ and $\beta_J^1$ in Stage 4. Note also that the distinguisher can use $\mathcal{A}'$'s valid defense $(c_J, \tau_{\mathrm{Rec}}^J)$ to compute one of the two sender's inputs, namely $s_J^{c_J}$. For the other input, $D'$ uses $s$, i.e. it sets $s_J^{1\oplus c_J} = s$ and completes the experiment using these inputs. Finally, $D'$ invokes $D$ on $\mathcal{A}$'s view and outputs whatever $D$ outputs. It follows from the construction that the view on which $D$ is invoked is distributed identically to $\mathcal{A}$'s view in $\widetilde{\mathsf{hyb}}_{e-1}(n)$ if $s$ is the sender's other input, namely, $s_J^{1\oplus c_J}$. On the other hand, if $s$ is a random string then the view is distributed identically to $\widetilde{\mathsf{hyb}}_e(n)$.

$$D'(\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n, (U_0^n, U_1^n)), \mathcal{A}(1^n)], U_{1-b}^n)) = D(\widetilde{\mathsf{hyb}}_{e-1}(n))$$

and

$$D'(\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n, (U_0^n, U_1^n)), \mathcal{A}(1^n)], \bar{U}^n)) = D(\widetilde{\mathsf{hyb}}_e(n))$$

where $\Gamma(v, *) = (v, *)$ if $v$ contains a valid defense for $\mathcal{A}'$. From Equation 4, it follows that the difference is non-negligible and that $\mathcal{A}'$ and $D'$ contradict the defensible privacy of protocol $\widehat{\pi}_{\mathrm{OT}}$ with respect to a corrupted receiver. This concludes the proof of the claim. □

**Hybrid 2:** In this hybrid game there is no trusted party that computes functionality $\mathcal{F}_{\mathrm{OT}}$. Instead, we define a simulator $\mathcal{S}_2$ that is given the sender's real inputs $v_0$ and $v_1$. Furthermore, $\mathcal{S}_2$ uses these inputs in Stage 5 of the execution. Then we claim that the receiver's view in Hybrids 1 and 2 is statistically close because the probability that the receiver learns more than $10n$ shares for both $u = 0$ and $u = 1$ is negligible. More formally,

**Claim 3.4.** *The following two distribution ensembles are statistically indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{OT}}}_{\pi_{\mathrm{HYBRID}_1}, \mathcal{S}_1, \mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\mathrm{s}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{EXTCOM}}}_{\pi_{\mathrm{HYBRID}_2}, \mathcal{S}_2, \mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** This follows from the facts that $|\Delta| = 18n$, $|\Gamma| = n$ and $|\Phi| \leq n$ with overwhelming probability (relying on the proof of Claim 3.2), and that the masking values that are used in Stage 5 are independent of the input to the OT executions in Stage 4. Specifically, the overall number of shares that the receiver learns is bounded by $|\Delta - \Gamma - \Phi| + 2|\Gamma| + 2|\Phi| \leq 20n$, where the rest of the shares are perfectly hidden (as their masking strings are not used elsewhere in the protocol). □

**Hybrid 3:** In this game we define a simulator $\mathcal{S}_3$ that is identical to $\mathcal{S}_2$ except for the way the sender's message is generated in Stage 4. More precisely, for all $i \in \Delta - \Gamma - \Phi$ it modifies the way $\beta_i^{1\oplus u}$ is computed. Recalling that $\mathcal{S}_2$ sets it to be random, then $\mathcal{S}_3$ will instead set

$$\beta_i^{1\oplus u} = \rho_i^{1\oplus u} \oplus s_i^{1\oplus c_i}.$$

Indistinguishability of hybrids 2 and 3 follows using the same proof as in Claim 3.3. Therefore, we have that the following ensembles are computationally indistinguishable.

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{OT}}}_{\pi_{\mathrm{HYBRID}_2},\mathcal{S}_2,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\mathrm{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{EXTCOM}}}_{\pi_{\mathrm{HYBRID}_3},\mathcal{S}_3,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

Observe that hybrid 3 is identical to the real execution. This concludes the proof of Claim 3.2. □

**Simulating the corrupted sender:** In this case $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ picks $20n$ random strings $a^1_{\mathrm{Rec}}, \ldots, a^{20n}_{\mathrm{Rec}}$ and emulates the ideal functionality $\mathcal{F}_{\mathrm{EXTCOM}}$ by invoking $20n$ times the commitment phase of $\omega_R$ with inputs $a^1_{\mathrm{Rec}}, \ldots, a^{20n}_{\mathrm{Rec}}$, against $\mathcal{A}$ that plays the role of receiver for the commitment scheme. At the end of this phase $\mathcal{S}$ obtains the output $((\widetilde{\mathsf{trans}_1}, \gamma_1), \ldots, (\widetilde{\mathsf{trans}_{20n}}, \gamma_{20n}))$ and receives from the adversary $20n$ random strings $b^1_{\mathrm{Rec}}, \ldots, b^{20n}_{\mathrm{Rec}}$. Next, $\mathcal{S}$ emulates functionality $\mathcal{F}_{\mathrm{EXTCOM}}$ in Stage 1 and invokes $20n$ times the commitment scheme $\omega_L$ with $\mathcal{A}$ (that plays the role of the committer), obtaining $((\widetilde{\mathsf{trans}'_1}, a^1_{\mathrm{Sen}}), \ldots, (\widetilde{\mathsf{trans}'_{20n}}, a^{20n}_{\mathrm{Sen}}))$. It internally records $a^1_{\mathrm{Sen}}, \ldots, a^{20n}_{\mathrm{Sen}}$ and further picks $20n$ random strings $b^1_{\mathrm{Sen}}, \ldots, b^{20n}_{\mathrm{Sen}}$, forwarding them to the adversary. The simulator also computes $r^i_{\mathrm{Sen}} = a^i_{\mathrm{Sen}} \oplus b^i_{\mathrm{Sen}}$ and then views $r^i_{\mathrm{Sen}} = s^0_i || s^1_i || \tau^i_{\mathrm{Sen}}$ $(s^0_i, s^1_i)$ is the input an honest sender must use in the $i^{th}$ OT protocol execution in Stage 3, together with randomness $\tau^i_{\mathrm{Sen}}$.

2. In Stage 2 the simulator participates with the adversary in $20n$ executions of the OT protocol $\widehat{\pi}_{\mathrm{OT}}$, while playing the role of the honest receiver. Note that due to the fact that the simulator knows the values of the input and randomness that the honest sender must use in each of the OT executions, the simulator can identify the coordinates of which the sender deviates. We denote this set of coordinates by the set $\Phi$.

3. In Stage 3 $\mathcal{S}$ receives $(q^1_{\mathrm{Sen}}, \ldots, q^n_{\mathrm{Sen}})$ from the sender and decommits the subset of values that corresponds to the coordinates in $\Gamma_{\mathrm{Sen}}$ as determined by $(q^1_{\mathrm{Sen}}, \ldots, q^n_{\mathrm{Sen}})$, playing the role of the receiver. Next, the simulator picks $n$ random numbers $(q^1_{\mathrm{Rec}}, \ldots, q^n_{\mathrm{Rec}})$ from $\{1, \ldots, 20\}^n$ and sends them to the sender. Upon receiving the decommitments from the sender, the simulator verifies the decommitments as would the honest receiver do with respect to $(q^1_{\mathrm{Rec}}, \ldots, q^n_{\mathrm{Rec}})$ and aborts in case of a mismatch. Finally, it samples a subset $\Gamma$ from $[20n]$ of size $n$ and emulates functionality $\mathcal{F}_{\mathrm{EXTCOM}}$ by invoking the commitment scheme $\omega_R$ with $\mathcal{A}$ (that plays the role of the receiver) $n$ times on input $\Gamma$, obtaining $((\mathsf{trans}'_1, \gamma_1), \ldots, (\mathsf{trans}'_{20n}, \gamma_{20n}))$. Let $\Delta = [20n] - \Gamma_{\mathrm{Rec}} - \Gamma_{\mathrm{Sen}} - \Phi$.

4. In Stage 4 the simulator proceeds as the honest receiver would do with input $u = 0$ and extracts the sender's inputs $v_0, v_1$. Specifically, the simulator knows all the inputs $\{(s^0_i, s^1_i)\}_{i\in\Delta}$ of the sender to the OT executions in Stage 2 and extracts the two sets of shares $\{\rho^0_i\}_{i\in\Delta}$ and $\{\rho^1_i\}_{i\in\Delta}$.

5. In Stage 5 the simulator plays the role of the honest receiver and checks whether the inputs and randomness revealed by the sender are consistent with the OT session that correspond to $\Delta \cap \Gamma$. In case of a mismatch the simulator aborts. Next, the simulator checks that $\tilde{\rho_0}$ and $\tilde{\rho_1}$ agree with some respective codewords $w_0$ and $w_1$ on $16n$ locations. In case of a non-agreement the simulator records a default value, else it records the codewords $w_0$ and $w_1$. It then runs a second consistency check to verify whether these codewords agree with $\beta^u_j \oplus s^{u\oplus\alpha_j}_j$ for all coordinates $j \in \Gamma$. If not, it records a default value. Finally, the simulator sends the recorded values to $\mathcal{F}_{\mathrm{OT}}$.

We next prove the following,

**Claim 3.5.** *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{F}_{\text{EXTCOM}}}_{\pi^{\text{ML}}_{\text{OT}},\mathcal{A},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\text{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\text{OT}}}_{\pi_{\text{IDEAL}},\mathcal{S},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** The security argument proceeds in a sequence of hybrids games starting from the simulated execution towards the real execution. We denote by $\pi_{\text{HYBRID}_i}$ the receiver's view in the $i^{th}$ hybrid game.

**Hybrid 1:** In this hybrid game there is no trusted party that computes functionality $\mathcal{F}_{\text{OT}}$. Instead, we define a simulator $\mathcal{S}_1$ that is given the receiver's real input $u$ and proceeds identically as $\mathcal{S}$ except for the way it generates the receiver's message in Stage 4. More precisely, $\mathcal{S}_1$ uses the real input $u$ instead of 0 in order to compute $\alpha_i$ for all $i \in \Delta - \Phi$. Indistinguishability of the simulation from the view in hybrid 1 follows from the receiver privacy of the OT protocol. More formally, we prove the following claim.

**Claim 3.6.** *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{F}_{\text{OT}}}_{\pi_{\text{IDEAL}},\mathcal{S},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\text{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\text{OT}}}_{\pi_{\text{HYBRID}_1},\mathcal{S}_1,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** Recall that the only difference between hybrid 1 and the simulated view is in the way that the messages in Stage 4 are generated. Specifically, in the simulated view $\mathcal{S}$ uses $u = 0$ in all sessions to compute $\alpha_i$. On the other hand, in game hybrid 1 the receiver uses the real $u$. Clearly, if the real input equals 0 then the views are identical and the proof of the claim follows immediately. Therefore, it suffices to consider the case $u = 1$. Towards proving this claim, we introduce a sequence of intermediate hybrid experiments $H^e_1$ for $e = 0, \ldots, 20n$. Namely, in hybrid $H^e_1$ we consider a simulator $\mathcal{S}^e$ that proceeds identically to $\mathcal{S}$ with the exception that it uses $u = 1$ in the first $e$ sessions in $\Delta$ in order to compute $\alpha_i$. By construction we have that the experiment $H^0_1$ proceeds identically to the ideal simulation and $H^{20n}_1$ proceeds identically to hybrid 1. Denote the view output in hybrid $H^e_1$ by $\mathsf{hyb}_e(n)$ and assume by contradiction that there exist a distinguisher $D$, a polynomial $p(\cdot)$ and infinitely many $n$'s such that

$$|\Pr[D(\mathbf{View}^{\mathcal{F}_{\text{OT}}}_{\pi_{\text{IDEAL}},\mathcal{S},\mathcal{Z}}(n)) = 1] - \Pr[D(\mathbf{View}^{\mathcal{F}_{\text{OT}}}_{\pi_{\text{HYBRID}_1},\mathcal{S}_1,\mathcal{Z}}(n)) = 1]| \geq \frac{1}{p(n)}\,.$$

Using a standard hybrid argument, it follows that there exists an $e \in \{1, \ldots, 20n\}$ such that

$$|\Pr[D(\mathsf{hyb}_e(n)) = 1] - \Pr[D(\mathsf{hyb}_{e-1}(n)) = 1]| \geq \frac{1}{20np(n)}\,.$$

As in the proof of Claim 3.3, we consider experiments $\overline{H}_{e-1}$ and $\overline{H}_e$ where the simulator samples a random $J \in [3n+1]$ and aborts if $J$ is not the $e^{th}$ session in $\Delta$. Next, we consider modified hybrids $\widetilde{H}_{e-1}$ and $\widetilde{H}_e$, where we slightly modify the receiver's messages in the coin-tossing phase and ask the receiver to commit to the all zeros string of appropriate length instead of committing to a uniform string $a^J_{\text{Rec}}$. Recalling that $a^J_{\text{Rec}}$ and $b^J_{\text{Rec}}$ determine the receiver's input in the $J^{th}$ execution of the OT protocol, we instruct the receiver to commit to 0 so that in the reduction we explain next we can forward the $J^{th}$'s execution messages to an external OT receiver in the reduction described next. More precisely, $\widetilde{H}_{e-1}$ (resp., $\widetilde{H}_e$) follows exactly as $\overline{H}_{e-1}$ (resp., $\overline{H}_e$) with the exception that, we modify the honest receiver's message in the coin-tossing stage, where it commits to the all zeros string instead of $a^j_{\text{Rec}}$. Denote by the random variables $\widetilde{\mathsf{hyb}}_e(n)$ and $\widetilde{\mathsf{hyb}}_{e-1}(n)$ the views of adversary

$\mathcal{A}$ in $\widetilde{H}_{e-1}$ and $\widetilde{H}_e$, respectively. Then following the same proof as in Claim 3.3, we can conclude that there exists a polynomial $q(\cdot)$ such that

$$\left| \Pr[D(\widetilde{\mathsf{hyb}}_e(n)) = 1] - \Pr[D(\widetilde{\mathsf{hyb}}_{e-1}(n)) = 1] \right| \geq \frac{1}{q(n)} \tag{5}$$

Without loss of generality, assume that

$$\Pr[D(\widetilde{\mathsf{hyb}}_e(n)) = 1] - \Pr[D(\widetilde{\mathsf{hyb}}_{e-1}(n)) = 1] \geq \frac{1}{q(n)} \, .$$

We use the above to construct a malicious sender $\mathcal{A}'$ that violates the privacy of the receiver relative to the oblivious transfer protocol $\widehat{\pi}_{\text{OT}}$. Specifically, $\mathcal{A}'$ internally emulates the experiment $\widetilde{H}_1^{e-1}$ by running the simulation strategy of $\mathcal{S}_1^{e-1}$ with the malicious sender $\mathcal{A}$, except for the following difference. $\mathcal{A}'$ relays the messages of the sender in the $J^{th}$ execution of the oblivious transfer protocol from Stage 2 to an external receiver with input $c$. Following the oblivious transfer executions, it continues the internal emulation until Stage 4. If $J$ is not the $e^{th}$ session in $\Delta$ then $\mathcal{A}'$ follows hybrid $\widetilde{H}_1^{e-1}$ and sets the view as $\perp$. Otherwise, sets $\alpha_J$ to be a random bit and continues the internal emulation to completion. It then invokes the distinguisher $D$ on $\mathcal{A}$'s internally generated view; denote by $b$ the bit output by $D$. Then, $\mathcal{A}'$ outputs $\alpha_j \oplus b$.

We proceed by analyzing the probability that $\mathcal{A}'$ correctly guesses $c$. Conditioned on not outputting fail and $\alpha_j \neq c$, the experiment emulated internally by $\mathcal{A}$ is identical to $\widetilde{H}_1^e$. Analogously, conditioned on not outputting fail and $\alpha_j = c$, the experiment emulated internally by $\mathcal{A}$ is identical to $\widetilde{H}_1^{e-1}$, where the probability that $\alpha_j = c$ (and $\alpha_j \neq c$) is $\frac{1}{2}$. Therefore,

$$\begin{aligned}
&\Pr[\ \mathcal{A}' \text{ guesses } c \text{ correctly }] \\
&\quad = \frac{1}{2} \Pr[D(\widetilde{\mathsf{hyb}}_e(n)) \oplus \alpha_j = c \mid \alpha_j \neq c\ ] \\
&\qquad\quad + \frac{1}{2} \Pr[D(\widetilde{\mathsf{hyb}}_{e-1}(n)) \oplus \alpha_j = c \mid \alpha_j = c\ ] \\
&\quad = \frac{1}{2} \Pr[D(\widetilde{\mathsf{hyb}}_e(n)) = 1] + \frac{1}{2} \Pr[D(\widetilde{\mathsf{hyb}}_{e-1}(n)) = 0] \\
&\quad = \frac{1}{2} + \frac{1}{2} \left( \Pr[D(\widetilde{\mathsf{hyb}}_e(n)) = 1] - \Pr[D(\widetilde{\mathsf{hyb}}_{e-1}(n)) = 1] \right) \\
&\quad \geq \frac{1}{2} + \frac{1}{2} \left( \frac{1}{q(n)} \right) .
\end{aligned}$$

Thus, we arrive at a contradiction. □

Note that the only difference between the real execution and hybrid 1 is in the way that the receiver outputs $v_u$. Specifically, in hybrid 1 simulator $\mathcal{S}_1$ extracts $v_0, v_1$ and then outputs $v_u$, while in the real execution the receiver outputs the value that corresponds to its strategy in Stage 5. We now prove that the receiver's output in both experiments is statistically close. In more details, the difference between the simulation's strategy and the honest receiver's strategy is that the simulator extracts the sender's both inputs in all $i \in \Delta - \Phi$ and then finds codewords that are $16n$-close to the extracted values, whereas the honest receiver finds a codeword that is $17n$-close based on the inputs it received in the Stages 2 and 5, and returns it.

Observe that the sender's views in hybrid 1 and the real execution are identical. It is therefore suffices to show that the value $u$ extracted by the simulator and fed to $\mathcal{F}_{\text{OT}}$ is identical the to the reconstructed output of the honest receiver. Let $v$ denote the value the honest receiver outputs and $v_u$ denote the value extracted by the simulator. These values are obtained in two steps:

- The honest receiver obtains shares of $v$ by computing $\tilde{\rho}_i = \beta_i^u \oplus \tilde{s}_i$ for $i \in \Delta$ where $\tilde{s}_i$ are its output from the OT sessions in Stage 2. On the other hand, the simulator computes $\overline{\rho}_i^u = \beta_j^u \oplus s_j^{u \oplus \alpha_j}$ where $s_j^b$'s are the inputs that the simulator extracted from Stage 1. (Note that these were the inputs that the sender was supposed to use in the OT sessions).
- Next, the closest codeword is computed from the shares. The honest receiver picks that code $\tilde{w}$ that is $17n$-close to $(\tilde{\rho}_i)_{i \in \Delta}$. The simulator, on the other hand, picks a codeword $\overline{w}^u$ that is $16n$-close to $(\overline{\rho}_i^u)_{i \in \Delta}$.

We now show that $v \neq v_u$ only with negligible probability because of the final cut-and-choose stage. We consider two cases:

**Case 1:** *The honest receiver extracts a valid $v$ from $(\tilde{\rho}_i)_{i \in \Delta}$*: In this case, we know that there is a codeword $w$ that is $17n$-close to $(\tilde{\rho}_i)_{i \in \Delta}$. Now, for every $i \in \Delta - \Phi$, we have that $\tilde{\rho}_i = \overline{\rho}_i$ since the sender proceeded honestly in those sessions. Following the same proof as in Claim 3.6, we can show that $|\Phi| \geq n$ only with negligible probability. Therefore, $|\Delta - \Phi| \geq 17n$ and $\tilde{\rho}_i$ and $\overline{\rho}_i^u$ agree on at least $17n$ locations in $\Delta - \Phi$. Now, since $w$ is $17n$-close to $(\tilde{\rho}_i)_{i \in \Delta}$, this means that $w$ is $16n$-close to $(\overline{\rho}_i^u)_{i \in [20n]}$ (because $|\Delta| = 18n$). Therefore the simulator would have recovered the same codeword and extracted the same value.

**Case 2:** *The honest receiver does not extract a valid $v$ from $(\tilde{\rho}_i)_{i \in \Delta}$*: This happens when $(\tilde{\rho}_i)_{i \in \Delta}$ is not $17n$-close to any codeword. In this case, the receiver uses a default value for $v$. We need to show that in this case, even the simulation sets $\tilde{v}_u$ to a default value. Suppose that, there exists $w$ that is $16n$-close to $(\overline{\rho}_i^u)_{i \in [20n]}$. We will argue that the simulator still sets $v_u$ to a default value.

Let $\psi$ be the locations where $w$ and $(\tilde{\rho}_i)_{i \in \Delta}$ differ. By our hypothesis that $(\tilde{\rho}_i)_{i \in \Delta}$ is not $17n$-close to any codeword, we have that $|\psi| > n$. Nevertheless, since $\Gamma$ is a randomly chosen subset of size $n$ and based on the proof of Claim 3.6, we can show that $\psi \cap \Gamma \neq \emptyset$ except with negligible probability. In this case, there exists an index $j \in \psi \cap \Gamma$ such that the sender must reveal values $s_j^0, s_j^1$ that are consistent with the OT protocol in session $i$ in Stage 2. Therefore, for such a $j \in \psi \cap \Gamma$ it holds that

$$\tilde{\rho_j} = \beta_j^u \oplus \tilde{s}_i = \beta_j^u \oplus s_j^{u \oplus \alpha_j} \neq w_j.$$

This implies that the simulator would have noted that $\beta_j^u \oplus s_j^{u \oplus \alpha_j} \neq w_j$. In this case the sender fails the second consistency check and the simulator should report that $\tilde{v}_u$ as the default value.

This concludes the proof of the claim. □

# 4 One-Sided Adaptive UC Secure Computation

In the two-party one-sided adaptive setting, at most one of the parties is adaptively corrupted [KO04, HP14]. In this section we provide a simple transformation of our static UC secure protocol from Section 3 to a two-party UC-secure protocol that is secure against one-sided adaptive corruption. Our first observation is that in Protocol 1 the parties use their real inputs to the OT protocol only in Phase 4. Therefore simulation of the first three phases can be easily carried out by simply following the honest strategy. On the other hand, simulating messages in Phase 4 requires some form of equivocation since if corruption takes place after this phase is concluded then the simulator needs to explain this message with respect to the real input of the corrupted party. On a high-level we will transform the protocol so that if no party is corrupted until end of Phase 4, the simulator can equivocate the message in Phase 4. We explain how to achieve equivocation later. First, we describe our simulator: In case either party is statically corrupted the simulation for Protocol 1 follows the strategy of the honest party until Phase 4, where the simulator extracts the corrupted party's

input relying on the fact that it knows the adversary's committed input in Phase 1. Therefore, the same proof follows in case the adversary adaptively corrupts one of the parties at any point before Phase 4, as the simulator can pretend that corruption took place statically. On the other hand, if corruption takes place after Phase 4, then the simulator equivocates the communication. It is important to note that while in the plain model any statically secure protocol can be compiled into one-sided secure protocol by encrypting its entire communication, it is not clear that this is the case in the UC setting due to the additional setup, e.g., a CRS that depends on the identity of the corrupted party. Nevertheless, in Phase 4 the parties only run a combiner for which the computation does not involve any usage of the CRS (which is induced by the extractable commitment). Therefore, the proof follows directly.

A common approach to achieve equivocation is to rely on non-committing encryption schemes (NCE) [CFGN96, DN00, CDMW09a], that allow secure communication in the presence of adaptive attacks. This powerful tool has been constructed while relying on (a variant of) simulatable PKE schemes, which, roughly speaking, allows for both the public-key and the ciphertexts to be generated obliviously without the knowledge of the plaintext or the secret key [DN00, CDMW09a]. Notably, these constructions achieve a stronger notion of security where both parties may be adaptively corrupted (also referred to as *fully adaptive*). Our second observation is that in the context of achieve one-sided adaptive security, it is sufficient to rely on a weaker variant of NCE, namely, one that is secure against only one-sided adaptive corruption.

In particular, we take advantage of a construction presented in [CFGN96] and later refined in [DN00], that achieves one-sided equivocation under the assumption of semi-honest OT with receiver equivocation. We will briefly describe it now. Recall that in the fully adaptive case, the high-level idea is for the sender and receiver to mutually agree on a random bit, which is then used by the sender to determine which of two random strings to mask its message. The process of agreeing on a bit requires the ability to both obliviously sample a public-key without the knowledge of the secret key, as well as the ability to obliviously sample a ciphertext without the knowledge of the corresponding plaintext. In the simpler one-sided scenario, Canetti et al. observed that an oblivious transfer protocol can replace the oblivious generation of the public-key. Specifically, the NCE receiver sends two public keys to the sender, and then the parties invoke an OT protocol where the NCE receiver plays the role of the OT sender and enters the corresponding secret keys. To allow equivocation for the NCE sender, the OT must enable equivocation with respect to the OT receiver. The [EGL85] OT protocol is an example for such a protocol. Here the OT receiver can pick the two ciphertexts so that it knows both plaintexts. Then equivocation is carried out by declaring that the corresponding ciphertext is obliviously sampled.

The advantage of this approach is that it removes the requirement of generating the public key obliviously, as now the randomness for its generation is split between the parties, where anyway only one of them is corrupted. This implies that the simulator can equivocate the outcome of the protocol execution without letting the adversary the ability to verify it. To conclude, it is possible to strengthen the security of Protocol 1 into the one-sided setting by simply encrypting the communication within the combiner phase using one-sided NCE which in turn can be constructed based on PKE with oblivious ciphertext generation. This implies the following theorem which further implies black-box one-sided UC secure computation from enhanced trapdoor permutation.

**Theorem 4.1.** *Assume the existence of PKE with oblivious ciphertext generation. Then for any two-party well-formed functionality $\mathcal{F}$, there exists a protocol that UC realizes $\mathcal{F}$ in the presence of one-sided adaptive, malicious adversaries in the CRS model using black-box access to the PKE.*

# 5 Adaptive UC Secure Computation

In this section we demonstrate the feasibility of UC secure commitment schemes based on PKE with oblivious ciphertext generation (namely, where it is possible to obliviously sample the ciphertext without knowing

the plaintext). Our construction is secure even in the presence of adaptive corruptions and is the first to achieve the stronger notion of adaptive security based on this hardness assumption. Plugging-in our UC commitment protocol into the transformation of [CDMW09b] that generates adaptive malicious OT given adaptive semi-honest OT and UC commitments, implies an adaptively UC secure oblivious transfer protocol with malicious security based on semi-honest adaptive OT and PKE with oblivious ciphertext generation using only black-box access to the semi-honest OT and the PKE. Stating formally,

**Theorem 5.1.** *Assume the existence of adaptive semi-honest oblivious transfer and PKE with oblivious ciphertext generation. Then for any multi-party well-formed functionality $\mathcal{F}$, there exists a protocol that UC realizes $\mathcal{F}$ in the presence of adaptive, malicious adversaries in the CRS model using black-box access to the oblivious transfer protocol and the PKE.*

Noting that simulatable PKE implies both semi-honest adaptive OT [CLOS02, CDMW09a] and PKE with oblivious ciphertext generation, we derive the following corollary (where simulatable PKE implies oblivious sampling of both public keys and ciphertexts),

**Corollary 5.2.** *Assume the existence of simulatable PKE. Then for any multi-party well-formed functionality $\mathcal{F}$, there exists a protocol that UC realizes $\mathcal{F}$ in the presence of adaptive, malicious adversaries in the CRS model using black-box access to the simulatable PKE.*

This in particular improves the result from [DMRV13] that relies on simulatable PKE in a non-black-box manner. Note also that our UC commitment can be constructed using a weaker notion than simulatable PKE where the inverting algorithms can require a trapdoor. This notion is denoted by trapdoor simulatable PKE [CDMW09a] and can be additionally realized based on the hardness assumption of factoring Blum integers. This assumption, however, requires that we modify our commitment scheme so that the CRS includes $3n + 1$ public keys of the underlying PKE instead of just one, as otherwise the reduction to the security of the PKE does not follow for multiple ciphertexts. Specifically, at the cost of linear blowup (in the security parameter) of the CRS, we obtain adaptively secure UC commitments under a weaker assumption. Now, since trapdoor simulatable PKE implies adaptive semi-honest OT [CDMW09a] it holds,

**Corollary 5.3.** *Assume the existence of trapdoor simulatable PKE. Then for any multi-party well-formed functionality $\mathcal{F}$, there exists a protocol that UC realizes $\mathcal{F}$ in the presence of adaptive, malicious adversaries in the CRS model using black-box access to the trapdoor simulatable PKE.*

Note that, since the best known general assumptions for realizing adaptive semi-honest OT is trapdoor simulatable PKE, this corollary gives evidence that the assumptions for adaptive semi-honest OT are sufficient for adaptive UC security and makes a step towards identifying the minimal assumptions for achieving UC security in the adaptive setting. To conclude, we note that enhanced trapdoor permutations, which imply PKE with oblivious ciphertext generation, imply the following corollary,

**Theorem 5.4.** *Assume the existence of enhanced trapdoor permutation. Then $\mathcal{F}_{\mathrm{COM}}$ (cf. Figure 2) can be UC realized in the CRS model in the presence of adaptive malicious adversaries.*

## 5.1 UC Commitments from PKE with Oblivious Ciphertext Generation

In this section we demonstrate the feasibility of adaptively secure UC commitments for the message space $m \in \{0,1\}$ from any public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Enc}}^{-1})$ with oblivious ciphertext generation (cf. Definition 2.4) in the common reference string (CRS) model. In this model [CF01] the parties have access to a CRS chosen from a specified trusted distribution $\mathcal{D}$. This is captured via the ideal functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ (see Figure 5). We note that we use $\Pi$ in two places in our protocol. First, in the encoding

**Protocol $\pi_{\mathrm{COM}}$.**

**CRS:** Two independent keys $\mathsf{PK}, \widetilde{\mathsf{PK}}$ that are in the range of $\mathsf{Gen}(1^n)$.

**Sender's Input:** A message $m \in \{0,1\}$ and a security parameter $1^n$.

**[Commitment phase:]**

**Encoding phase:** The sender chooses a random $n$-degree polynomial $p(\cdot)$ over a field $\mathbb{F}[x]$ such that $p(0) = m$. Namely, it randomly chooses $a_i \leftarrow \mathbb{F}$ for all $i \in [n]$ and sets $a_0 = m$, and defines the polynomial $p(x) = a_0 + a_1 x + \cdots + a_n x^n$. The sender then creates a commitment to $m$ as follows. For every $i = [3n+1]$, it first pick $b_i \leftarrow \{0,1\}$ at random and then computes the following pairs:

$$\text{If } b_i = 0 \text{ then } \begin{array}{rcl} c_i^0 & = & \mathsf{Enc}_{\mathsf{PK}}(p(i); t_i) \\ c_i^1 & = & r_i \end{array} \quad \text{else,} \quad \text{if } b_i = 1 \text{ then } \begin{array}{rcl} c_i^0 & = & r_i \\ c_i^1 & = & \mathsf{Enc}_{\mathsf{PK}}(p(i); t_i) \end{array}$$

where $t_i \leftarrow \{0,1\}^n$ and $r_i \leftarrow \widetilde{\mathsf{Enc}}(\cdot)$ is obliviously sampled. The sender sends $(c_0^0, c_0^1), \ldots, (c_{3n+1}^0, c_{3n+1}^1)$ to the receiver.

**Coin-tossing phase:** The sender and receiver interact in a coin-tossing protocol that is carried out as follows.

1. The receiver sends $c = \mathsf{Enc}_{\widetilde{\mathsf{PK}}}(\sigma_0; r_{\sigma_0})$ to the sender where $\sigma_0 \leftarrow \{0,1\}^N$ is chosen uniformly at random.
2. The sender picks $\sigma_1 \leftarrow \{0,1\}^N$ at random and sends it in the clear to the receiver
3. The receiver decrypts $c$ by revealing $\sigma_0$ and $r_{\sigma_0}$.

Both the sender and the receiver compute $\sigma = \sigma_0 \oplus \sigma_1$ and use $\sigma$ as the random string to sample a random subset $S \subset [3n+1]$ of size $n$. (Note that such sampling can be done in a simple way by partitioning the set of coordinates into $n$ sets of triples (where the last set includes $4$ elements) and picking one element per set. Notably, this technique does not imply that any potential subset of size $n$ will be picked, rather it ensures that a subset is picked with a negligible probability in $n$, specifically $(1/3)^n$, which suffices for our proof.)

**Cut-and-choose phase:** The sender decrypts the set $\{c_i^{b_i}\}_{i \in S}$ by sending the sequence $\{b_i, p(i), t_i\}_{i \in S}$. The receiver verifies that all the decryptions are correct and aborts otherwise.

**[Decommitment phase:]** Let $T = [3n+1] - S$. The sender reveals its input $m$ and decrypts all the ciphertexts in $\{c_i^{b_i}\}_{i \in T}$. The receiver checks if all the decryptions are correct and aborts otherwise. Using the $n$ polynomial evaluations revealed relative to $i \in S$ and any additional polynomial evaluation that was revealed relative to $T$, the receiver reconstructs the polynomial $p(\cdot)$ (via polynomial interpolation of $n+1$ points). Next, the receiver verifies whether $p(0) = m$, and that for every $i \in [3n+1]$ the point $p(i)$ is the decrypted value within $c_i^{m_i}$.

Figure 4: UC adaptively secure commitment scheme.

phase (where the commitments are computed by the sender) and then in the coin-tossing phase (where the commitments are computed by the receiver). Our complete construction can be found in Figure 4. Next, we prove that

**Theorem 5.5.** *Assume that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Enc}}^{-1})$ is a PKE with oblivious ciphertext generation. Then protocol $\pi_{\mathrm{COM}}$ (cf. Figure 4) UC realizes $\mathcal{F}_{\mathrm{COM}}$ in the CRS model in the presence of adaptive malicious adversaries.*

**A high level proof.** Intuitively, security requires proving both hiding and binding in the presence of static and adaptive corruptions. The hiding property follows from the IND-CPA security of the encryption scheme combined with the fact that the receiver only sees $n$ shares in a $n$-out-of-$3n+1$ secret-sharing of the message in the commit phase. On the other hand, proving binding is much more challenging and reduces to the facts

that a corrupted sender cannot successfully predict exactly the $n$ indices from $\{1, \ldots, 3n+1\}$ that will be chosen in the coin-tossing protocol. In fact, if it can identify these $n$ indices, then it would be possible for the adversary to break binding. An important information-theoretic argument that we prove here is that for a fixed encoding phase, no adversary can equivocate on two continuations from the encoding phase with different outcomes of the coin-tossing phase. Saying differently, for any given encoding phase there is exactly one outcome for the coin-tossing phase that will allow equivocation. Given this claim, binding now follows from the IND-CPA security of the encryption scheme used in the coin-tossing phase.

In addition, recall that in the UC setting the scheme must also support a simulation that allows straight-line extraction and equivocation. At a high-level, the simulator sets the CRS to public-keys for which it knows the corresponding secret-keys. This will allow the simulator to extract all the values encrypted by the adversary. We observe that the simulator can fix the outcome of the coin-tossing phase to any $n$-indices of its choice by extracting the random string $\sigma_0$ encrypted by the receiver and choosing a random string $\sigma_1$ so that $\sigma_0 \oplus \sigma_1$ is a particular string. Next, the simulator generates secret-sharing for both 0 and 1 so that they overlap in the particular $n$ shares. To commit, the simulator encrypts the $n$ common shares within the $n$ indices to be revealed (which it knows in advance), and for the rest of the indices it encrypts two shares, one that corresponds to the sharing of 0 and the other that corresponds to the sharing of 1. Finally, in the decommit phase, the simulator reveals that shares that correspond to the real message $m$, and exploits the invertible sampling algorithm to prove that the other ciphertexts were obliviously generated.

## 5.2 Proof of Theorem 5.5

Let $\mathcal{A}$ be a malicious probabilistic polynomial-time real adversary running the above protocol in the $\mathcal{F}_{\text{CRS}}$-hybrid model. We construct an ideal model adversary $\mathcal{S}$ with access to $\mathcal{F}_{\text{COM}}$ which simulates a real execution of protocol $\pi_{\text{COM}}$ with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal process with $\mathcal{S}$ and $\mathcal{F}_{\text{COM}}$ from a real execution of $\pi_{\text{COM}}$ with $\mathcal{A}$. $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with environment $\mathcal{Z}$, emulating the honest party. We separately describe the actions of $\mathcal{S}$ for every corruption case.

**Initialization:** The common reference string (CRS) is chosen by $\mathcal{S}$ in the following way. It generates $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^n)$ and $(\widetilde{\text{PK}}, \widetilde{\text{SK}}) \leftarrow \mathsf{Gen}(1^n)$, and places $(\text{PK}, \widetilde{\text{PK}})$ in the CRS. The simulator further records $(\text{SK}, \widetilde{\text{SK}})$.

**Simulating the communication with $\mathcal{Z}$:** Every message that $\mathcal{S}$ receives from $\mathcal{Z}$ it internally feeds to $\mathcal{A}$ and every output written by $\mathcal{A}$ is relayed back to $\mathcal{Z}$.

**Simulating the commitment phase when the receiver is statically corrupted:** In this case $\mathcal{S}$ proceeds as follows:

1. *Encoding phase:* Upon receiving message $(sid, \text{Sen}, \text{Rec})$ from $\mathcal{F}_{\text{COM}}$, the simulator picks a random subset $S^* \subset [3n+1]$ of size $n$ and two random $n$-degree polynomials $p_0(\cdot)$ and $p_1(\cdot)$ such that:

$$\begin{aligned} p_0(i) &= p_1(i) \qquad \forall i \in S^* \\ p_0(0) = 0 \quad &\text{and} \quad p_1(0) = 1. \end{aligned}$$

   Note that the simulator can define these polynomials via interpolation, where a unique $n$-degree polynomial can be constructed given $n+1$ points. Let $T^* = [3n+1] - S^*$, then the simulator defines the commitment as follows:

- For every $i \in S^*$ the simulator proceeds as the honest sender would with polynomial $p_0(\cdot)$. Namely, it first picks $b_i \leftarrow \{0,1\}$ at random and then sets the following pairs,

$$\text{If } b_i = 0 \text{ then } \begin{array}{rcl} c_i^0 &=& \mathsf{Enc}_{\mathsf{PK}}(p_0(i);t_i) \\ c_i^1 &=& r_i \end{array} \quad \text{else, if } b_i = 1 \text{ then } \begin{array}{rcl} c_i^0 &=& r_i \\ c_i^1 &=& \mathsf{Enc}_{\mathsf{PK}}(p_0(i);t_i) \end{array}$$

  where $b_i \leftarrow \{0,1\}$, $t_i \leftarrow \{0,1\}^n$ and $r_i \leftarrow \widetilde{\mathsf{Enc}_{\mathsf{PK}}}(\cdot)$ is obliviously sampled (we recall that $p_0(i) = p_1(i)$ for all $i \in S^*$).

- For every $i \in T^*$ the simulator picks $b_i \leftarrow \{0,1\}$ at random and then uses the points on both polynomials $p_0(\cdot)$ and $p_1(\cdot)$ to calculate the following pairs,

$$\text{If } b_i = 0 \text{ then } \begin{array}{rcl} c_i^0 &=& \mathsf{Enc}_{\mathsf{PK}}(p_0(i);t_i) \\ c_i^1 &=& \mathsf{Enc}_{\mathsf{PK}}(p_1(i);\tilde{t}_i) \end{array} \quad \text{else, if } b_i = 1 \text{ then } \begin{array}{rcl} c_i^0 &=& \mathsf{Enc}_{\mathsf{PK}}(p_1(i);\tilde{t}_i) \\ c_i^1 &=& \mathsf{Enc}_{\mathsf{PK}}(p_0(i);t_i) \end{array}$$

  where $b_i \leftarrow \{0,1\}$ and $t_i, \tilde{t}_i \leftarrow \{0,1\}^n$ are chosen uniformly at random.

Finally, the simulator sends the pairs $(c_0^0, c_0^1), \ldots, (c_{3n+1}^0, c_{3n+1}^1)$ to the receiver.

2. *Coin-tossing phase:* The simulator biases the coin-tossing result so that the set $S$ that is chosen in this phase is identical to $S^*$. More precisely, the simulator extracts $\sigma_0$ from the receiver's ciphertext and then sets $\sigma_1$ so that $\sigma = \sigma_0 \oplus \sigma_1$ yields the set $S^*$.
3. *Cut-and-choose phase:* The simulator decrypts all the ciphertexts within $\{c_i^{b_i}\}_{i \in S^*}$.

**Simulating the decommitment phase where the receiver is statically corrupted:** Upon receiving a message $(\mathsf{reveal}, sid, m)$ from $\mathcal{F}_{\mathrm{COM}}$, $\mathcal{S}$ generates a simulated decommitment message as follows. Recall first that the simulator needs to reveal points on a polynomial $p(\cdot)$ and pairs $\{(b_i, t_i)\}_{i \in [3n+1]}$ such that $p(0) = m$ and $c_i^{b_i} = \mathsf{Enc}_{\mathsf{PK}}(p(i);t_i)$. Let $\hat{b}_i = b_i \oplus m$ for all $i \in T^*$, then $\mathcal{S}$ reveals $p_m(\cdot)$, $\{\hat{b}_i, t_i^{\hat{b}_i}, r_i = \widetilde{\mathsf{Enc}_{\mathsf{PK}}}^{-1}(c_i^{1-\hat{b}_i})\}_{i \in T^*}$.

Next, we prove that $\mathcal{Z}$ cannot distinguish an interaction of protocol $\pi_{\mathrm{COM}}$ with $\mathcal{A}$, corrupting the receiver, from an interaction of $\mathcal{S}$ with $\mathcal{F}_{\mathrm{COM}}$. Formally,

**Claim 5.1.** *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{CRS}}}_{\pi_{\mathrm{COM}},\mathcal{A},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\mathrm{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{COM}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** We prove this claim using a sequence of hybrid games.

**Hybrid** $0$**:** This is the real interaction of $\mathcal{Z}$ with $\mathcal{A}$ and Protocol $\pi_{\mathrm{COM}}$.

**Hybrid** $1$**:** In this experiment we define a simulator $\mathcal{S}_1$ that proceeds as follows. $\mathcal{S}_1$ uses $\mathcal{S}$'s strategy in the coin-tossing phase when simulating the corrupted receiver. Specifically, $\mathcal{S}_1$ emulates $\mathcal{F}_{\mathrm{CRS}}$ and generates $(\mathrm{PK}, \mathrm{SK})$ and $(\widetilde{\mathrm{PK}}, \widetilde{\mathrm{SK}})$ as in the simulation. Next, it picks at the beginning of the commit phase a random subset $S^*$ for which it wishes to bias the outcome of the coin-tossing phase. It then extracts the value $\sigma_0$ encrypted by the receiver in the coin-tossing phase using $\widetilde{\mathrm{SK}}$, and sets $\sigma_1$ so that $\sigma_0 \oplus \sigma_1$ results in $S^*$. $\mathcal{S}_1$ next follows the honest's sender role with input $m$ for the rest of the execution. We claim that the adversary's view in this hybrid game is identically distributed to its view in the prior hybrid. This is because $S^*$ is chosen independently of the hybrid game and uniformly at random. Therefore, given any particular $\sigma_0$ extracted from the adversary's commitment in the

coin-tossing stage, $\sigma_1$ will be uniformly random (which is exactly how it is distributed in hybrid 0). Therefore, we have that the following distributions are identical,

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{CRS}}}_{\pi_{\mathrm{COM}},\mathcal{A},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \equiv \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{COM}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S}_1,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Hybrid** 2**:** In this experiment, we define a simulator $\mathcal{S}_2$ that is given the sender's message $m$, yet it carries out $\mathcal{S}$'s strategy in the encoding phase instead of playing the role of the honest sender. More precisely, $\mathcal{S}_2$ proceeds identically to $\mathcal{S}_1$ with the exception that in the encoding phase, it defines polynomials $p_m(\cdot)$ and $p_{1-m}(\cdot)$ exactly as $\mathcal{S}$ does in the simulation using the set $S^*$. Observe first that the outcome of the coin-tossing phase has already been fixed to $S^*$ in hybrid 1. Moreover, $\mathcal{S}_2$ executes the decommitment phase exactly as the honest sender does by providing polynomial $p_m(\cdot)$. Then the differences between the receiver's view in hybrids 1 and 2 are with respect to the non-opened ciphertexts. Namely, the ciphertexts that are in positions $1-b_i$'s, and denoted by $\{c_i^{1-b_i}\}_{i\in[3n+1]}$, which encode the polynomial $p_{1-m}(\cdot)$. These ciphertexts are obliviously picked in hybrid 1, yet computed using algorithm $\mathsf{Enc}$ in hybrid 1. We now prove that the receiver's views in these hybrids executions are computational indistinguishability due to the indistinguishability of ciphertexts using $\mathsf{Enc}$ and $\widetilde{\mathsf{Enc}}$. More precisely, we show the following claim:

**Claim 5.2.** *The following distributions are computationally indistinguishable.*

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{COM}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S}_1,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\mathrm{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{COM}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S}_2,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** Assume by contradiction that there exist a PPT adversary $\mathcal{A}$, distinguisher $D$ and polynomial $p(\cdot)$ such that $D$ can distinguish the two distributions stated in the claim for infinitely many $n$'s with probability $\frac{1}{p(n)}$. Fix an $n$ for which this happens. Then we use $\mathcal{A}$ we construct an adversary $\mathcal{A}'$ that violates the indistinguishability of real and obliviously generated ciphertexts (cf. definition 2.4). Towards this, we consider a sequence of intermediate hybrid games $H_2^0,\ldots,H_2^{3n+1}$ where in hybrid $H_2^j$ we define a simulator $\mathcal{S}_2^j$ that proceeds identically to $\mathcal{S}_2$ when generating $\{c_i^{1-b_i}\}_{i\in[j]}$, namely, it picks a polynomial $p_{1-m}(\cdot)$ and sets $c_i^{1-b_i}$ by $\mathsf{Enc}_{\mathsf{PK}}(p_{1-m}(i))$. Finally, $\{c_i^{1-b_i}\}_{i>j}$ are obliviously generated as in the real sender's strategy. Note that by our construction, $H_2^0$ and $H_2^{3n+1}$ proceed identically to Hybrids 1 and 2, respectively. Denoting the output of the execution in hybrid $H_2^j$ by $\mathsf{hyb}_j(n)$ and using a standard hybrid argument, it follows that there exists $j$ such that,

$$\left|\Pr[D(\mathsf{hyb}_j(n))=1]-\Pr[D(\mathsf{hyb}_{j-1}(n))=1]\right| \geq \frac{1}{(3n+1)p(n)}. \tag{6}$$

We now construct an adversary $\mathcal{A}'$ that violates the indistinguishability of obliviously generated ciphertexts and real ciphertexts. Specifically, recall that $\mathcal{A}'$ needs to distinguish $(\mathrm{PK},r_1,c_1)$ from $(\mathrm{PK},r_1',c_2)$ where $c_1 \leftarrow \widetilde{\mathsf{Enc}}_{\mathsf{PK}}(r_1)$ and $c_2 \leftarrow \mathsf{Enc}_{\mathsf{PK}}(m;r_2), r_1' \leftarrow \widetilde{\mathsf{Enc}}^{-1}(c_2)$. Upon receiving $(\mathrm{PK},r,c)$ $\mathcal{A}'$ proceeds as follows. It first emulates the execution as in hybrid 1 by setting the CRS to be $(\mathrm{PK},\widetilde{\mathrm{PK}})$ for $(\widetilde{\mathrm{PK}},\widetilde{\mathrm{SK}}) \leftarrow \mathsf{Gen}(1^n)$. It then emulates the internal execution by following the strategy of $\mathcal{S}_1^{j-1}$ with the exception that $c_j^{1-b_j}$ is set to $c$. Later, when $\mathcal{A}'$ needs to reveal $c_j^{1-b_j}$ it returns $r$ as the randomness used to obliviously generate $c$. Finally, $\mathcal{A}'$ invokes $D$ on $\mathcal{A}$'s view and outputs whatever $D$ outputs. We recall that the ciphertexts that correspond to the set $\{1-b_i\}_{i\in[3n+1]}$ are always revealed as obliviously generated ciphertexts regardless of the way they were generated. It must also be noted that $\mathcal{A}'$ does not need to know $\mathrm{SK}$ in order to complete the simulation of the

sender's messages since it never extracts here. Nevertheless, $\mathcal{A}'$ does need access to $\widetilde{\mathsf{Enc}}^{-1}$ in order to generate the randomness of the first $j-1$ ciphertexts, which by the definition of the encryption scheme requires only PK. To conclude, the internal emulation of $\mathcal{A}'$ upon receiving $(\mathrm{PK}, r_1, c_1)$ so that $c_1 \leftarrow \widetilde{\mathsf{Enc}}_{\mathrm{PK}}(r_1)$ is identically distributed to $H_j$, whereas when $(\mathrm{PK}, r'_1, c_2)$ are generated so that $c_2 \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; r_2), r'_1 \leftarrow \widetilde{\mathsf{Enc}}^{-1}_{\mathrm{PK}}(c_2)$ with $m = p_{1-m}(j)$, $\mathcal{A}$'s view is distributed identically to $H_{j-1}$. Therefore, it follows from Equation 6 that

$$\Big|\Pr[\mathcal{A}'(\mathrm{PK}, r_1, c_1) = 1] - \Pr[\mathcal{A}'(\mathrm{PK}, r'_1, c_2) = 1]\Big| \geq \frac{1}{(3n+1)p(n)}.$$

This implies a contradiction relative to the indistinguishability property of real and obliviously generated ciphertexts. □

**Hybrid** 3**:** This hybrid is the actual simulation with $\mathcal{S}$. Namely, here $\mathcal{S}_3$ does not have the honest sender's actual input $m$ and it computes two polynomials $p_0(\cdot)$ and $p_1(\cdot)$ as defined above. Furthermore, $\mathcal{S}_3$ reveals one of the polynomials $p_0(\cdot)$ or $p_1(\cdot)$ in the decommitment phase, depending on the value of $m$. Observe that the distribution of the messages sent by $\mathcal{S}_2$ and $\mathcal{S}_3$ is identical in both hybrids. We use the facts that at most $n$ shares are revealed in the commitment phase and that $p(\cdot)$ is an $n$-degree polynomial. Therefore, revealing these $n$ shares keeps $p_{1-m}(0)$ completely hidden and we have that,

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{COM}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S}_2,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \equiv \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{COM}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S}_3,\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

□

**Simulating the commit phase when the sender is statically corrupted:** Simulating the sender involves extracting the committed value as follows:

1. *Encoding phase:* The simulator proceeds honestly following the honest receiver's strategy, receiving pairs $(c_i^0, c_i^1)$ for all $i \in [3n+1]$. The simulator exploits the fact that it knows the secret key SK and decrypts all ciphertexts. Let $\beta_i^b = \mathsf{Dec}_{\mathrm{SK}}(c_i^b)$.
2. *Coin-tossing phase:* The simulator proceeds honestly following the honest receiver's strategy. Let $S'$ be the outcome of the coin-tossing phase.
3. *Cut-and-choose phase:* The simulator proceeds as the honest receiver and verifies if the openings are consistent with the ciphertexts sent in the encoding phase. Note that none of the revealed values should differ from what the simulator decrypted using SK due to the fact that $\Pr[\mathsf{Dec}_{\mathrm{SK}}(\mathsf{Enc}_{\mathrm{PK}}(m)) = m] = 1$.
4. *Input extraction:* Finally, the simulator extracts the sender's input as follows. $\mathcal{S}$ chooses an arbitrary index $j \in [3n+1] - S'$ and reconstructs two polynomials $q(\cdot)$ and $\widetilde{q}(\cdot)$ such that

$$q(i) = \widetilde{q}(i) = \beta_i^{b_i} \quad \forall i \in S'$$
$$q(j) = \beta_j^0 \quad \text{and} \quad \widetilde{q}(j) = \beta_j^1 \quad \text{and} \quad q(0), \widetilde{q}(0) \in \{0,1\}.$$

It then verifies whether for all $i \in [3n+1]$, $q(i) \in \{\beta_i^0, \beta_i^1\}$ and $\widetilde{q}(i) \in \{\beta_i^0, \beta_i^1\}$. The following cases arise:

**Case 1:** *Both $q(\cdot)$ and $\widetilde{q}(\cdot)$ satisfy the condition and $\widetilde{q}(0) \neq q(0)$.* Then $\mathcal{S}$ halts returning fail. Below we prove that the simulator outputs fail with negligible probability.

**Case 2:** *At most one of $q(\cdot)$ and $\widetilde{q}(\cdot)$ satisfy the condition or $\widetilde{q}(0) = q(0)$.* $\mathcal{S}$ sends $(\mathsf{commit}, sid, q(0))$ to the $\mathcal{F}_{\mathrm{COM}}$ functionality and stores the committed bit $q(0)$. Otherwise, $\mathcal{S}$ sends a default value.

**Case 3:** *Neither $q(\cdot)$ or $\widetilde{q}(\cdot)$ satisfy the condition.* $\mathcal{S}$ sends a default value to the ideal functionality and need not store the committed bit since it will never be decommitted correctly.

**Claim 5.3.** *Conditioned on case* 1 *not occurring, the sender can decommit to* $b$ *if and only if* $\mathcal{S}$ *sends* $b$ *to* $\mathcal{F}_{\mathrm{COM}}$.

*Proof.* By the assumption in the claim, either case 2 or 3 occur. We now show that if $\mathcal{A}$ decommits successfully, then it must be either with polynomial $q(\cdot)$ or $\widetilde{q}(\cdot)$ if both satisfy the conditions, or with the single satisfying polynomial. That would imply that the adversary can only decommit to whatever sent by the simulator to $\mathcal{F}_{\mathrm{COM}}$. We will demonstrate our argument for the case that both polynomials satisfy the condition. The case of a single polynomial follows similarly. More formally, suppose that $q(\cdot)$ and $\widetilde{q}(\cdot)$ are as required by the above condition. Then polynomial $q^*(\cdot)$ that is revealed by $\mathcal{A}$ in the decommitment phase must take the same value as $q(\cdot)$ and $\widetilde{q}(\cdot)$ for all $i \in S'$. Focusing on the $j^{th}$ index that is specified above, it holds that either $q^*(j) = q(j)$ or $q^*(j) = \widetilde{q}(j)$ (because $\mathcal{A}$ can decrypt ciphertexts $c_j^0$ and $c_j^1$ only to the plaintexts $q(j)$ and $\widetilde{q}(j)$, respectively). This implies that either $q^*(\cdot)$ and $q(\cdot)$ share $n+1$ points or $q^*(\cdot)$ and $\widetilde{q}(\cdot)$ share $n+1$ points. Consequently, $q^*(\cdot)$ becomes identically equal to either $q(\cdot)$ or $\widetilde{q}(\cdot)$ since it is an $n$-degree polynomial, and $\mathcal{A}$ can only decommit to $q(0)$ or $\widetilde{q}(0)$. ■

**Claim 5.4.** *The probability that* $\mathcal{S}$ *outputs* fail *in case* 1 *is negligible.*

*Proof.* Assume for contradiction that there exists $\mathcal{A}$ that for infinitely many $n$'s generates ciphertexts for the encoding phase such that $\mathcal{S}$ obtains valid $q_0(\cdot)$ and $q_1(\cdot)$ such that both satisfy the conditions at the end of the commit phase with probability at least $\frac{1}{p(n)}$. Observe that in such a case, the transcript can be equivocated to both 0 and 1 using $q_0(\cdot)$ and $q_1(\cdot)$, respectively. We show how to construct $\mathcal{A}'$ that violates the privacy of the underlying encryption scheme. At a high-level, we prove that $\mathcal{A}$ can successfully equivocate only if it biases the coin-tossing outcome, and this can be achieved only by breaking the privacy of the encryption scheme.

We consider first an alternative simulator $\widetilde{\mathcal{S}}$ that proceeds exactly as the real simulator $\mathcal{S}$ does, with the exception that it receives as input $\mathrm{PK}^*$ that it internally sets $\widetilde{\mathrm{PK}}$ in the CRS. Observe that $\mathcal{S}$ does not use the $\widetilde{\mathrm{SK}}$ in simulating the corrupted sender. Hence the view generated by $\widetilde{\mathcal{S}}$ is identical to $\mathcal{S}$. This implies that the transcript that is obtained in the simulated commit phase of $\widetilde{\mathcal{S}}$ can be equivocated with probability $\frac{1}{p(n)}$, i.e., there are valid decommitments to both 0 and 1 relative to polynomials $q_0(\cdot)$ and $q_1(\cdot)$. Then, by applying a standard averaging argument it holds that the transcript from the commit phase can be equivocated with probability $\frac{1}{2p(n)}$ over a random continuation of some fixed transcript $\tau$, where $\tau$ is a partial transcript of protocol $\pi_{\mathrm{COM}}$ that reaches right after the encoding phase and the probability is taken over the adversary's and honest receiver's randomness. (Specifically, the probability that $\frac{1}{2p(n)}$ portion of the partial transcripts lead to a successful equivocation is $\frac{1}{2p(n)}$.) Using this observation we will construct an adversary $\mathcal{B}$ that wins the IND-CPA game for the encryption scheme $\Pi$.

Our proof relies heavily of the following claim,

**Claim 5.5.** *Let* $\tau$ *be a fixed partial transcript as above. Then there exist no transcripts* $\mathsf{trans}_1, \mathsf{trans}_2$ *that satisfy the following conditions:*

1. $\mathsf{trans}_1$ *and* $\mathsf{trans}_2$ *are complete and accepting transcripts of* $\pi_{\mathrm{COM}}$ *with* $\tau$ *being their prefix.*
2. *There exists two distinct sets* $S_1, S_2$ *such that* $S_1$ *and* $S_2$ *are the respective outcomes of the coin-tossing phase within* $\mathsf{trans}_1$ *and* $\mathsf{trans}_2$.

3. *There are valid decommitments to values* $0$ *and* $1$ *in* $\mathsf{trans}_1$ *and* $\mathsf{trans}_2$.

An important observation that follows from Claim 5.5 is that the sets chosen in the coin-tossing phase must be identical for any two complete $\mathsf{trans}_1, \mathsf{trans}_2$ that are defined relative to a fixed partial transcript $\tau$, on which a transcript can be equivocated. Clearly, given that the receiver's random string $\sigma_0$ in the coin-tossing phase is hidden from the sender, it must hold that the probability that the same set is chosen in two coin-tossing executions is exponentially small. On the other hand, with non-negligible probability over partial transcripts $\tau$, there are decommitments to both $0$ and $1$, and from the Claim 5.5 we know that a successful equivocation implies a fixed joint set $S^*$ of size $n$. This intuitively means that the adversary violates the IND-CPA security of the encryption scheme $\Pi$. Formally, we construct an adversary $\mathcal{B}$ that internally incorporates $\mathcal{A}$ and proceeds as in the IND-CPA game:

1. $\mathcal{B}$ externally receives a public-key $\mathrm{PK}^*$. It follows $\widetilde{\mathcal{S}}$'s strategy and sets $\widetilde{\mathrm{PK}}$ in the CRS as this input.
2. $\mathcal{B}$ emulates an execution with $\mathcal{A}$ following $\widetilde{\mathcal{S}}$'s strategy until the completion of the encoding phase. Denote the partial transcript obtained so far by $\tau$.
3. $\mathcal{B}$ samples $M_1 = np(n)$ transcripts with prefix $\tau$ as follows. It invokes $\mathcal{A}$, $M_1$ times, each time with independent randomness for $\widetilde{\mathcal{S}}$ (which specifically implies independent randomness in the coin-tossing phase). For each such execution $\mathcal{B}$ checks whether there are two valid decommitments for $0$ and $1$. If there exists one such transcript, $\mathcal{B}$ stores the outcome $S^*$ of the coin-tossing phase on that transcript. If no such transcript is encountered, $\mathcal{B}$ outputs a random bit and halts.
4. $\mathcal{B}$ samples two random strings $\sigma_0^0$ and $\sigma_0^1$ independently at random from $\{0,1\}^N$ and outputs these strings. Upon receiving a ciphertext $c$ from its oracle, $\mathcal{B}$ feeds $c$ internally as the receiver's message in the coin-tossing phase within the partial transcript $\tau$. It then invokes $\mathcal{A}$ on $(\tau, c)$ and completes the execution as follows. If $\mathcal{A}$ aborts then $\mathcal{B}$ outputs a random bit and halts. Otherwise, let $\sigma$ be the string revealed by $\mathcal{A}$ in the coin-tossing phase. If $\sigma \oplus \sigma_0^{b'}$ does not result in $S^*$ as the outcome of the coin-tossing for some $b' \in \{0,1\}$, then $\mathcal{B}$ outputs a random bit. Otherwise $\mathcal{B}$ outputs $b'$ and halts. (Note that in any case $\mathcal{B}$ aborts the execution right before it needs to decrypt $c$, since it cannot do that).

We will now prove that $\mathcal{B}$ successfully identifies whether $c$ is an encryption of $\sigma_0^0$ or $\sigma_0^1$ with probability non-negligibly greater than $\frac{1}{2}$. Towards proving that we consider the following events, conditioned on $c$ being an encryption of $\sigma_0^b$:

$\mathrm{E}_1$**:** There are decommitments to $0$ and $1$ conditioned on transcript $\tau$ that is generated in Step 2. We already argued above that the probability that $\mathrm{E}_1$ occurs is at least $\frac{1}{2p(n)}$.

$\mathrm{E}_2$**:** Here we consider the event that $\mathcal{B}$ successfully computes $S^*$ in Step 3. Note first that the probability a transcript, generated in Step 3, fails to reveal $S^*$ is at most $1 - \frac{1}{2p(n)}$. This is due to the fact that set $S^*$ can be efficiently extracted whenever there are two decommitments. Therefore,

$$\Pr[\mathrm{E}_1 \mid \mathrm{E}_2] = 1 - \left(1 - \frac{1}{2\mathrm{p}(\mathrm{n})}\right)^{\mathrm{np(n)}} = 1 - \mathrm{e}^{-\mathrm{n}/2}.$$

$\mathrm{E}_3$**:** Finally, we consider the event that the coin-tossing phase results in $S^*$. Note that by Claim 5.5 whenever the transcript can be equivocated then it must be that the coin-tossing result is $S^*$, this

implies that

$$\begin{aligned}\Pr[\mathrm{E}_3 \mid \mathrm{E}_2 \ \wedge \ \mathrm{E}_1] &\geq \Pr[\text{ transcript with prefix } \tau \text{ can be equivocated} \mid \mathrm{E}_2 \ \wedge \ \mathrm{E}_1] \\ &\geq \frac{1}{2p(n)}.\end{aligned}$$

From the definition of the events it follows that if $\mathrm{E}_1 \wedge \mathrm{E}_2 \wedge \mathrm{E}_3$ occurs, then $\mathcal{B}$ wins the IND-CPA game with probability 1. Denote the joint events $\mathrm{E}_1 \wedge \mathrm{E}_2 \wedge \mathrm{E}_3$ by Comp and note that if Comp does not occur then $\mathcal{B}$'s guess is correct with probability $\frac{1}{2}$. Then from the calculation above it holds that,

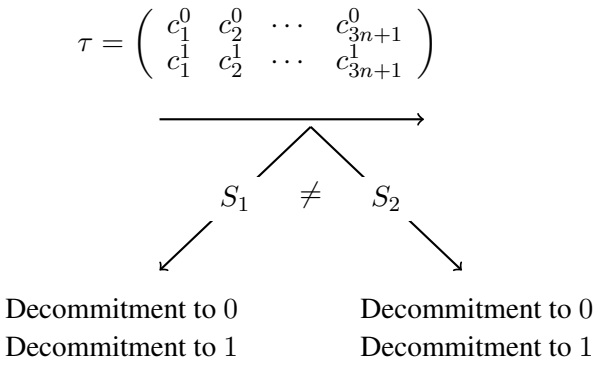
$$\begin{aligned}\Pr[\text{Comp}] &= \Pr[\mathrm{E}_3 \wedge \mathrm{E}_2 \wedge \mathrm{E}_1] = \Pr[\mathrm{E}_3 \mid \mathrm{E}_2 \wedge \mathrm{E}_1] \cdot \Pr[\mathrm{E}_2 \mid \mathrm{E}_1] \cdot \Pr[\mathrm{E}_1] \\ &\geq \frac{1}{2p(n)} \times (1 - e^{-n/2}) \times \frac{1}{2p(n)} \\ &\geq \frac{1}{8(p(n))^2}.\end{aligned}$$

Next, we compute the probability that $\mathcal{B}$ succeeds in its guess.

$$\begin{aligned}\mathrm{ADV}_{\Pi,\mathcal{B}}(n) &= \Pr[\mathcal{B} \text{ succeeds } |\text{Comp}] \cdot \Pr[\text{Comp}] + \Pr[\mathcal{B} \text{ succeeds } |\neg\text{Comp}] \cdot \Pr[\neg\text{Comp}] \\ &= 1 \cdot \Pr[\text{Comp}] + \frac{1}{2}\Pr[\neg\text{Comp}] \\ &= 1 \cdot \Pr[\text{Comp}] + \frac{1}{2}(1 - \Pr[\text{Comp}]) \\ &= \frac{1}{2} + \frac{1}{2}\Pr[\text{Comp}] \\ &\geq \frac{1}{2} + \frac{1}{16(p(n))^2}.\end{aligned}$$

This concludes the proof of Claim 5.6. It remains to prove Claim 5.5.

**Proof of Claim 5.5.** Assume for contradiction there exists a partial transcript $\tau$ of the encoding phase, complete transcripts $\mathsf{trans}_1$ and $\mathsf{trans}_2$ and sets $S_1 \neq S_2$ as in Claim 5.5. Let PK be the public-key in the CRS and SK be the corresponding secret-key. We define some notations first.




- We denote by $T_1 = [3n+1] - S_1$ and $T_2 = [3n+1] - S_2$.
- Recall that transcripts $\mathsf{trans}_1$ and $\mathsf{trans}_2$ include valid decommitments to both 0 and 1. Moreover, since we assume that the prefix $\tau$ is in common to both transcripts and the decryption is prefect, then a ciphertext $c_i^b$ that is decrypted in either $\mathsf{trans}_1$ or $\mathsf{trans}_2$ must be correctly revealed into exactly one plaintext, which is determined by $\beta_i^b = \mathsf{Dec}_{\mathsf{SK}}(c_i^b)$.

- By the assumption above, transcript $\mathsf{trans}_1$ induces two valid decommitments to both 0 and 1. We denote the $b_i$ values within the decommitment to 0 by $(b_1^0, \ldots, b_{3n+1}^0)$, and the values within the decommitment to 1 by $(b_1^1, \ldots, b_{3n+1}^1)$.
- Similarly, transcript $\mathsf{trans}_2$ induces two valid such decommitments. Then, let the $b_i$ values within the decommitment to 0 be denoted by $(\widehat{b}_1^0, \ldots, \widehat{b}_{3n+1}^0)$, and the values within the decommitment to 1 denoted by $(\widehat{b}_1^1, \ldots, \widehat{b}_{3n+1}^1)$.

Consider transcript $\mathsf{trans}_1$. Then the shares that correspond to the indices in $S_1$ and are revealed during the commitment phase imply that,

$$\beta_i^0 = \beta_i^1 \quad \forall\, i \in S_1.$$

This further imply that,

$$b_i^0 \;=\; b_i^1 \quad \forall\, i \in S_1. \tag{7}$$

The rest of the shares are revealed in the decommitment phase. Now, since we use an $(n+1)$-out-of-$(3n+1)$ secret sharing scheme, these $n$ shares that correspond to the indices in $S_1$, together with any additional revealed share $i \in T_1$, constitute $n+1$ shares from which a unique polynomial can be reconstructed. Specifically, the reconstructed polynomials for decommitting to 0 and 1 have to be different (since the secrets are different), so it must be that the revealed plaintexts are also different for every $i \in T_1$, i.e.

$$\beta_i^0 \neq \beta_i^1 \quad \forall\, i \in T_1.$$

This means that, for $i \in T_1$, $\{c_i^0, c_i^1\}$ must contain the plaintexts $\beta_i^0$ and $\beta_i^1$. Hence $(c_i^0, c_i^1)$ must either be the encryption of $(\beta_i^0, \beta_i^1)$ or $(\beta_i^1, \beta_i^0)$. In either case we have that

$$b_i^0 \;=\; 1 - b_i^1 \quad \forall\, i \in T_1. \tag{8}$$

Next, we consider transcript $\mathsf{trans}_2$ and recall that it shares the same encoding phase with $\mathsf{trans}_1$. It thus must be the case that for every $i \in [3n+1]$ the revealed shares for both transcripts must correspond to either $\beta_i^0$ or $\beta_i^1$. From Equation 8 we know that for every $i \in T_1$, $b_i^0 \neq b_1^1$. Hence, for every $i \in T_1$, either $\widehat{b}_i^0 = b_i^0$ or $\widehat{b}_i^0 = b_i^1$. Relying on the fact that $|T_1| = 2n+1$ and the pigeon-hole principle, it must hold that there are $n+1$ indices in $T_1$ for which either $\widehat{b}_i^0 = b_i^0$ for all these indices, or $\widehat{b}_i^0 = b_i^1$ for all these indices. This implies two cases:

$\widehat{b}_i^0 = b_i^0$ **on** $n+1$ **locations:** In this case, the revealed shares for these $n+1$ locations must be the same for both $\mathsf{trans}_1$ and $\mathsf{trans}_2$ when decommitting to 0. Note that if $n+1$ shares are the identical then the polynomials revealed for both transcripts must be identical as well, and therefore the revealed shares for every other index must be identical. We additionally know that the plaintext shares $\beta_i^0 \neq \beta_i^1$ for any index $i \in T_1$. Combining this with the fact that $\widehat{b}_i^0$ must correspond to the same share as the one corresponding to $b_i^0$, it follows that $\widehat{b}_i^0 = b_i^0$ for all $i \in T_1$.

$\widehat{b}_i^0 = b_i^1$ **on** $n+1$ **locations:** In this case we conclude, analogously to the previous case, that the polynomial revealed when decommitting to 0 in $\mathsf{trans}_2$ must be the same as when decommitting to 1 in $\mathsf{trans}_1$. However, such a decommitment on $\mathsf{trans}_2$ is invalid because the secret, which is the value of this polynomial evaluated at 0 is 1.

Therefore, we can conclude that it must be first case, where the revealed polynomials are identical so that $\widehat{b}_i^0 = b_i^0$ for all $i \in T_1$ (and not just for the $n+1$ locations). Applying the same argument, we can

prove that $\widehat{b}_i^1 = b_i^1$ for all $i \in T_1$. Now, since for every $i \in T_1$ we have that $b_i^0 = 1 - b_i^1$, it follows that:

$$\widehat{b}_i^0 = 1 - \widehat{b}_i^1 \quad \forall i \in T_1.$$

Next, we observe that $T_1 \cap S_2$ is non-empty since $S_1 \neq S_2$. We conclude with a contradiction by considering $i^* \in T_1 \cap S_2$. Specifically, for any such $i^*$ we have from the preceding argument that the fact that $i^* \in T_1$ implies that

$$\widehat{b}_{i^*}^0 \quad = \quad 1 - \widehat{b}_{i^*}^1$$

On the other hand, $i^* \in S_2$ and the values for all the indices in $S_2$ are already revealed in the commitment phase of $\mathsf{trans}_2$, thus we have that

$$\widehat{b}_{i^*}^0 = \widehat{b}_{i^*}^1$$

which is a contradiction. ■

**Simulating the decommit phase when the sender is statically corrupted:** $\mathcal{S}$ first checks whether the decommitted value is the value stored during the commit phase and whether the decommitment is valid. If these two conditions are met then $\mathcal{S}$ sends $(\mathsf{reveal}, sid)$ to $\mathcal{F}_{\mathrm{COM}}$. Otherwise, $\mathcal{S}$ ignores the message.

Next, we prove that $\mathcal{Z}$ cannot distinguish an interaction of protocol $\pi_{\mathrm{COM}}$ with $\mathcal{A}$, corrupting the sender, from an interaction of $\mathcal{S}$ with $\mathcal{F}_{\mathrm{COM}}$. Formally,

**Claim 5.6.** *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{F}_{\mathrm{CRS}}}_{\pi_{\mathrm{COM}},\mathcal{A},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \stackrel{\mathrm{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}_{\mathrm{COM}}}_{\pi_{\mathrm{IDEAL}},\mathcal{S},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Proof:** Here we need to argue that the simulator outputs fail with negligible probability and that the receiver outputs the same message in both executions. Recall first the event for which the simulator fails occurs when it computes the polynomials $q_0$ and $q_1$ and then finds out that both of them satisfy the condition in case 1. In this case, $\mathcal{A}$ can equivocate the committed message in the decommit phase. We thus prove that the probability that $\mathcal{A}$ can generate such polynomials is negligible, which implies that the probability that $\mathcal{S}$ fails is negligible. More precisely, we prove in the following lemma that the probability that $\mathcal{A}$ can break the binding property is negligible probability. A key point in our proof relies on the fact that a malicious sender cannot bias the coin-tossing outcome (as opposed to the simulator). Formally, □

**Simulating the commit phase when the sender is corrupted after the encoding phase:** Upon corruption, $\mathcal{S}$ receives the sender's input $m$. It then reveals the sender's randomness just as it would do when simulating a decommitment for an uncorrupted sender. Computational indistinguishability follows similarly to the case that the receiver is statically corrupted.

**Simulating the commit phase when the receiver is corrupted anywhere during the commit phase:** Recall that $\mathcal{S}$ honestly simulates the receiver messages, and thus it can simply reveal the randomness of the receiver.

# References


[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 186–195, 2004.

[Bea91] Donald Beaver. Foundations of secure interactive computing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 377–391, 1991.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145, 2001.

[CDMW09a] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 287–302, 2009.

[CDMW09b] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 387–402, 2009.

[CDPW06] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. *IACR Cryptology ePrint Archive*, 2006:432, 2006.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 19–40, 2001.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 639–648, 1996.

[CKL06] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pages 73–88, 2013.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503, 2002.

[CPS07] Ran Canetti, Rafael Pass, and Abhi Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 249–259, 2007.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 426–437, 2003.

[DMRV13] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkitasubramaniam. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 316–336, 2013.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 432–450, 2000.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 581–596, 2002.

[DNO10] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. On the necessary and sufficient assumptions for UC computation. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 109–127, 2010.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 325–335, 2000.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 51–60, 2012.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 505–523, 2009.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, pages 412–426, 2008.

[HIK+11] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM J. Comput.*, 40(2):225–266, 2011.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptology*, 25(1):158–193, 2012.

[HP14] Carmit Hazay and Arpita Patra. One-sided adaptively secure two-party computation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 368–393, 2014.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 99–108, 2006.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 572–591, 2008.

[IR88] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 8–26, 1988.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31, 1988.

[KLP07] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent composition of secure protocols in the timing model. *J. Cryptology*, 20(4):431–492, 2007.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 335–354, 2004.

[Lin03] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 394–403, 2003.

[Lin09] Yehuda Lindell. Adaptively secure two-party computation with erasures. In *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, pages 117–132, 2009.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 446–466, 2011.

[LP12] Huijia Lin and Rafael Pass. Black-box constructions of composable protocols without setup. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 461–478, 2012.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 179–188, 2009.

[LPV12] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for UC from only OT. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 699–717, 2012.

[LZ09] Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 183–201, 2009.

[MPR10] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. A zero-one law for cryptographic complexity with respect to computational UC security. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 595–612, 2010.

[MR91] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 392–404, 1991.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 448–457, 2001.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 554–571, 2008.

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 403–418, 2009.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Tech. Memo TR-81, Aiken Computation Laboratory, Harvard U., 1981.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.

# A UC Security

We briefly recall the Universal Composability (UC) framework, for more details we refer to [Can01].

**Environment.** The model of execution includes a special entity called the UC-environment (or environment) $\mathcal{Z}$. The environment "manages" the whole execution: it invokes all the parties at the beginning of the

execution, generates all inputs and reads all outputs, and finally produces an output for the whole concurrent execution. Intuitively, the environment models the "larger world" in which the concurrent execution takes place (e.g., for a distributed computing task over the Internet, the environment models all the other activities occurring on the Internet at the same time).

**Adversarial behavior.** The model of execution also includes a special entity called the adversary, that represents adversarial activities that are directly aimed at the protocol execution under consideration. We consider both *static* and *adaptive* adversaries, where static adversarial strategy implies corruptions at the onset of the protocol execution, whereas adaptive strategy implies corruptions at any point during the execution and as a function of what the attacker's view. When a party is corrupted, it shares all its tapes with the adversary and follows its instructions for all its future actions.

While honest parties only communicate with the environment through the input/output of the functions they compute, the adversary is also able to exchange messages with the environment in an arbitrary way throughout the computation.[5] Furthermore, the adversary controls the scheduling of the delivery of all messages exchanged between parties (where messages sent by the environment are directly delivered). Technically, this is modeled by letting the adversary read the outgoing message tapes of all parties and decide whether or not and when (if at all) to deliver the message to the recipient, therefore the communication is asynchronous and lossy. However, the adversary cannot insert messages and claim arbitrary sender identity. In other words, the communication is authenticated.

**Protocol execution.** The *execution of a protocol $\pi$ with the environment $\mathcal{Z}$, adversary $\mathcal{A}$ and trusted party $\mathcal{G}$* proceeds as follows. The environment is the first entity activated in the execution, who then activates the adversary, and invokes other honest parties. At the time an honest party is invoked, the environment assigns it a unique identifier and inquiries the adversary whether it wants to corrupt the party or not. To start an execution of the protocol $\pi$, the environment initiates a *protocol execution session*, identified by a session identifier $sid$, and activates all the participants in that session. An activated honest party starts executing the protocol $\pi$ thereafter and has access to the trusted party $\mathcal{G}$. We remark that in the UC model the environment only initiates one protocol execution session.

***Invoking parties.*** The environment invokes an honest party by passing input (invoke, $P_i$) to it. $P_i$ is the globally unique identity for the party and is picked dynamically by the environment at the time it is invoked. Immediately after that, the environment notifies the adversary of the invocation of $P_i$ by sending the message (invoke, $P_i$) to it, who can then choose to corrupt this party by replying (corrupt, $P_i$). Note that here as the adversary is static, parties are corrupted only when they are "born" (invoked).

***Session initiation.*** To start an execution of protocol $\pi$, the environment selects a subset $U$ of parties that has been invoked so far. For each party $P_i \in U$, the environment activates $P_i$ by sending a start-session message (start-session, $P_i$, $sid$, $c_{i,sid}$, $x_{i,sid}$) to it, where $sid$ is a session id that identifies this execution. We remark that in the UC model, the environment starts only one session, and hence all the activated parties have the same session id.

***Honest party execution.*** An honest party $P_i$, upon receiving (start-session, $P_i$, $sid$, $c_{i,sid}$, $x_{i,sid}$), starts executing its code $c_{i,sid}$ input $x_{i,sid}$. During the execution,

- The environment can read $P_i$'s output tape and at any time may pass additional inputs to $P_i$;
- According to its code, $P_i$ can send messages (delivered by the adversary) to the other parties in the session, in the format ($P_i$, $P_j$, $s$, content),[6] where $P_j$ is the identity of the receiver;

[5] Through its interaction with the environment, the adversary is also able to influence the inputs to honest parties indirectly.

[6] The session id in the messages enables the receiver to correctly de-multiplexing a message to its corresponding session, even though the receiver may involve in multiple sessions simultaneously.

- According to its code, $P_i$ can send an input to the trusted party in the format ($P_i$, $\mathcal{F}$, $s$, input).

***Adversary execution.*** Upon activation, the adversary may perform one of the following activities at any time during the execution.

- The adversary can read the outgoing communication tapes of all honest parties and decides to deliver some of the messages.
- The adversary can exchange arbitrary messages with the environment.
- The adversary can read the inputs, outputs and incoming messages of a corrupted party, and instruct the corrupted party for any action.
- The (adaptive) adversary can decide to corrupt any party from the set of honest parties at the moment.

***Output.*** The environment outputs a final result for the whole execution in the end.

In the execution of protocol $\pi$ with security parameter $n \in \mathbb{N}$, environment $\mathcal{Z}$, adversary $\mathcal{A}$ and trusted party $\mathcal{G}$, we define $\mathbf{View}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(n)$ to be the random variable describing the output of the environment $\mathcal{Z}$, resulting from the execution of the above procedure.

Let $\mathcal{F}$ be an ideal functionality; we denote by $\pi_{\text{IDEAL}}$ the protocol accessing $\mathcal{F}$, called as the ideal protocol. In $\pi_{\text{IDEAL}}$ parties simply interacts with $\mathcal{F}$ with their private inputs, and receive their corresponding outputs from the functionality at the end of the computation. Then the *ideal model execution* of the functionality $\mathcal{F}$ is just the execution of the ideal protocol $\pi_{\text{IDEAL}}$ with environment $\mathcal{Z}$, adversary $\mathcal{A}'$ and trusted party $\mathcal{F}$. The output of the execution is thus $\mathbf{View}^{\mathcal{F}}_{\pi_{\text{IDEAL}},\mathcal{A}',\mathcal{Z}}(n)$. On the other hand, the real model execution does not require the aid of any trusted party. Let $\pi$ be a multi-party protocol implementing $\mathcal{F}$. Then, the *real model execution* of $\pi$ is the execution of $\pi$ with security parameter $n$, environment $\mathcal{Z}$ and adversary $\mathcal{A}$, whose output is the random variable $\mathbf{View}_{\pi,\mathcal{A},\mathcal{Z}}(n)$. Additionally, the *$\mathcal{G}$-Hybrid model execution* of a protocol $\pi$ is the execution of $\pi$ with security parameter $n$, environment $\mathcal{Z}$ and adversary $\mathcal{A}$ and ideal functionality $\mathcal{G}$.

**Security as emulation of a real model execution in the ideal model.** Loosely speaking, a protocol *securely realizes* an ideal functionality if it *securely emulates* the ideal protocol $\pi_{\text{IDEAL}}$. This is formulated by saying that for every adversary $\mathcal{A}$ in the real model, there exists an adversary $\mathcal{A}'$ (a.k.a. *simulator*) in the ideal model, such that no environment $\mathcal{Z}$ can tell apart if it is interacting with $\mathcal{A}$ and parties running the protocol, or $\mathcal{A}'$ and parties running the ideal protocol $\pi_{\text{IDEAL}}$.

**Definition A.1. (UC security)** *Let $\mathcal{F}$ and $\pi_{\text{IDEAL}}$ be defined as above, and $\pi$ be a multi-party protocol in the $\mathcal{G}$-hybrid model. Then protocol $\pi$ UC realizes $\mathcal{F}$ with static (resp. adaptive) security in $\mathcal{G}$-hybrid model, if for every uniform* PPT *static (resp. adaptive) adversary $\mathcal{A}$, there exists a uniform* PPT*simulator $\mathcal{A}'$, such that for every non-uniform* PPT *environment $\mathcal{Z}$, the following two ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}} \overset{\text{c}}{\approx} \left\{\mathbf{View}^{\mathcal{F}}_{\pi_{\text{IDEAL}},\mathcal{A}',\mathcal{Z}}(n)\right\}_{n\in\mathbb{N}}.$$

**Multi-session extension of ideal functionalities.** Note that the UC model only considers a single session of the protocol execution. (Namely, the environment is only allowed to open one session). To consider multiple concurrent executions, we focus on the multi-session extension of ideal functionalities [Can01, CLOS02]. More specifically, let $\hat{\mathcal{F}}$ be the multi-session extension of $\mathcal{F}$. That is, $\hat{\mathcal{F}}$ runs multiple copies of $\mathcal{F}$, where each copy will be identified by a special "sub-session identifier". Every $k$ parties, trying access $\mathcal{F}$ together, share a sub-session identifier, $ssid$. To compute the function, each party simply sends its private input

together with $ssid$ to $\hat{\mathcal{F}}$. Upon receiving all the inputs, $\hat{\mathcal{F}}$ activates the appropriate copy of $\mathcal{F}$ identified by $ssid$ (running within $\hat{\mathcal{F}}$), and forwards the incoming messages to that copy. (If no such copy of $\mathcal{F}$ exists then a new copy is invoked and is given that ssid.) Outputs that are generated by the copies of $\mathcal{F}$ are returned to corresponding parties by $\hat{\mathcal{F}}$.

# B From bit OT to String OT

In this section, we demonstrate how to transform a bit OT protocol to a string OT protocol in a way that preserves both defensible private with respect to a corrupted receiver and sender privacy. At a high-level, in order to obtain an $n$-bits OT we repeat the bit OT protocol in parallel $n$-times. Given two $n$-bits strings $(s_0, s_1)$, the sender enters the $i^{th}$ bit of $s_0$ and $s_1$ as the bit-inputs for the $i^{th}$ OT execution. In addition, the receiver must use the same index for all executions. We next prove that sender privacy of the transformed protocol follows easily using a simple hybrid argument. Defensible privacy, on the other hand, holds since the receiver is required to produce a good defense for the $n$ parallel executions simultaneously and a good defense should show that the receiver supplied the same index in all the executions. We note that our proof works for random OT, which is sufficient for our purposes, yet can be extended for the more general case.

The definition of defensible privacy for $n$-bit OT is provided in Section 2.3. The definition for defensible privacy for a bit OT follows analogously and is identical to the definition presented in [HIK$^+$11]. Below we present an equivalent game-based security formulation for bit-OT defensible privacy, that is further described in [HIK$^+$11]. In what follows, we reduce the defensible privacy of the string OT according to Definition 2.7 to this game-based formulation of defensible privacy for bit OT. More formally, we consider the following experiment for a protocol $\pi$ and a PPT adversary $\mathcal{A}$:

**Experiment** $\mathsf{Expt}_{\pi}(\mathcal{A})$ :

1. Choose $s_0, s_1 \in \{0,1\}$ uniformly at random.
2. Let $\rho_{\mathrm{Sen}}$ be a uniformly chosen random tape for the sender $\mathrm{Sen}$, and let $\mathsf{trans}$ be a transcript of an interaction between the adversary $\mathcal{A}$ and $\mathrm{Sen}((s_0, s_1), \rho_{\mathrm{Sen}})$.
3. Let $((r, \rho_{\mathrm{Rec}}), s^*)$ be the output of $\mathcal{A}(1^n)$ on transcript $\mathsf{trans}$ where $(r, \rho_{\mathrm{Rec}})$ is a defense and $s^*$ is a guess for $s_{1-r}$.
4. Output 1 only if $(r, \rho_{\mathrm{Rec}})$ is a good defense for $\mathcal{A}$ in $\mathsf{trans}$ and $s^* = s_{1-r}$.

**Definition B.1.** *An OT protocol is defensible private with respect to a corrupted receiver if for any* PPT *adversary $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large $n$'s,*

$$\Pr[\mathsf{Expt}_{\pi}(\mathcal{A}(1^n)) = 1] \leq \frac{1}{2} + \mu(n).$$

We now provide our transformation and prove correctness. Let $\pi_{\mathrm{OT}}$ be a bit OT protocol. We construct a string OT protocol $\pi^n_{\mathrm{OT}}$ using $\pi_{\mathrm{OT}}$ as follows.

**Lemma B.1.** *Assume that $\pi_{\mathrm{OT}}$ is a bit OT protocol that is defensible private with respect to a corrupted receiver and sender private. Then $\pi^n_{\mathrm{OT}}$ is a string OT protocol that is defensible private with respect to a corrupted receiver and sender private.*

*Proof.* We first prove sender privacy of the string OT protocol.

**Sender privacy:** Loosely speaking, sender privacy requires that no malicious sender can distinguish the case when the receiver's input is 0 and 1, with non-negligible probability. Suppose for contradiction that

**Protocol 2** (Protocol $\pi_{\text{OT}}^n$)**.**

**Input:** *The sender* Sen *has input* $(v_0, v_1)$ *where* $v_0, v_1 \in \{0,1\}^n$ *and the receiver* Rec *has input* $u \in \{0,1\}$.

**The protocol:**
*The parties participate in* $n$ *executions of the OT protocol* $\pi_{\text{OT}}$ *where the receiver uses* $u$ *as its input in all executions and the sender uses the* $i^{th}$ *bits of* $v_0$ *and* $v_1$ *as its input in execution* $i$*.*

there exist a PPT adversary $\mathcal{A}$, distinguisher $D$ and polynomial $p(\cdot)$ such that $D$ distinguishes the following distributions with probability at least $\frac{1}{p(n)}$ for infinitely many $n$'s,

- $\{\textbf{View}_{\mathcal{A},\pi_{\text{OT}}^n}[\mathcal{A}(1^n), \text{Rec}(1^n, 0)]\}_{n\in\mathbb{N}}$,
- $\{\textbf{View}_{\mathcal{A},\pi_{\text{OT}}^n}[\mathcal{A}(1^n), \text{Rec}(1^n, 1)]\}_{n\in\mathbb{N}}$.

Fix an $n$ for which this happens. We construct $\mathcal{A}'$ and distinguisher $D'$ using $\mathcal{A}$ and $D$ that violates the sender privacy of $\pi_{\text{OT}}$. We introduce a sequence of intermediate hybrid experiments $H_0, \ldots, H_n$, where in hybrid $H_i$ we consider a receiver $\text{Rec}_i$ that follows the honest receiver's code in each of the $n$ parallel executions of $\pi_{\text{OT}}$ with the exception that it uses the input 1 in the first $i$ executions and 0 in the remaining executions. Let $\mathsf{hyb}_i(n)$ denote the view of the adversary in hybrid $H_i$. Then by construction we have that

$$\begin{aligned}\mathsf{hyb}_0(n) &= \textbf{View}_{\mathcal{A},\pi_{\text{OT}}^n}[\mathcal{A}(1^n), \text{Rec}(1^n, 0)], \text{and}\\ \mathsf{hyb}_n(n) &= \textbf{View}_{\mathcal{A},\pi_{\text{OT}}^n}[\mathcal{A}(1^n), \text{Rec}(1^n, 1)]\end{aligned}$$

Moreover, using a standard hybrid argument there exists $i$ such that $D$ distinguishes $\mathsf{hyb}_{i-1}(n)$ and $\mathsf{hyb}_i(n)$ with probability at least $\frac{1}{np(n)}$.

Then adversary $\mathcal{A}'$ is defined as follows. It internally emulates the hybrid experiment $\mathsf{hyb}_{i-1}(n)$ by playing the role of the honest rceeiver against $\mathcal{A}$, with the exception that it forwards $\mathcal{A}$'s messages in the $i^{th}$ execution to an external receiver. Consider the function $\mathsf{reconstruct}$ that on input the view of $\mathcal{A}'$ in an interaction using $\pi_{\text{OT}}$ reconstructs the view of $\mathcal{A}$ in the internal emulation of $\mathcal{A}'$. It follows from our construction that if the receiver uses input 0 in the interaction with $\mathcal{A}'$ this view of $\mathcal{A}$ is identically distributed to $\mathsf{hyb}_{i-1}(n)$. If the receiver uses input 1 the view is identically distributed to $\mathsf{hyb}_i(n)$. More precisely,

$$\begin{aligned}\mathsf{reconstruct}(\textbf{View}_{\mathcal{A}',\pi_{\text{OT}}}[\mathcal{A}(1^n), \text{Rec}(1^n, 0)]) &= \mathsf{hyb}_{i-1}(n)\\ \mathsf{reconstruct}(\textbf{View}_{\mathcal{A}',\pi_{\text{OT}}}[\mathcal{A}(1^n), \text{Rec}(1^n, 1)]) &= \mathsf{hyb}_i(n)\end{aligned}$$

Next, we construct a distinguisher $D'$. On input a view of $\mathcal{A}'$, runs the function $\mathsf{reconstruct}$ on the view and runs $D$ on the output of the function. Finally, $D'$ outputs what $D$ outputs. It now follows that

$$\begin{aligned}\Pr[D'(\textbf{View}_{\mathcal{A}',\pi_{\text{OT}}}[\mathcal{A}(1^n), \text{Rec}(1^n, 0)]) = 1] &= \Pr[D(\mathsf{hyb}_{i-1}(n)) = 1]\\ \Pr[D'(\textbf{View}_{\mathcal{A}',\pi_{\text{OT}}}[\mathcal{A}(1^n), \text{Rec}(1^n, 1)]) = 1] &= \Pr[D(\mathsf{hyb}_i(n)) = 1]\end{aligned}$$

which implies that $D'$ distinguishes $\mathcal{A}'$'s view when the receiver's input are 0 and 1 because $D$ distinguishes $\mathsf{hyb}_{i-1}(n)$ and $\mathsf{hyb}_i(n)$ with probability at least $\frac{1}{np(n)}$, which contradicts the sender privacy of $\pi_{\text{OT}}$.

**Defensible privacy with respect to a corrupted receiver:** As mentioned before, we reduce the security of defensible-privacy for string OT according to Definition 2.7 to the game-based formulation of defensible privacy for bit OT (cf. Definition B.1). Assume by contradiction that there exists an adversary $\mathcal{A}$ that violates the defensible privacy of $\pi_{\text{OT}}^n$ with respect to a corrupted receiver. More precisely, suppose there exist a PPT adversary $\mathcal{A}$, distinguisher $D$ and polynomial $p(\cdot)$ such that for infinitely many $n$'s, $D$ distinguishes the following distributions with probability at least $\frac{1}{p(n)}$,

- $\{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(U_0^n,U_1^n)),\mathcal{A}(1^n)],U_{1-b}^n)\}$, and
- $\{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(U_0^n,U_1^n)),\mathcal{A}(1^n)],\bar{U}^n)\}$

where $\Gamma(v,*)$ equals $(v,*)$ if when following the execution $\mathcal{A}$ outputs a good defense for $\pi$, and $\perp$ otherwise, $b$ is Rec's input in this defense and $U_0^n, U_1^n, \bar{U}^n$ are independent random variables that are uniformly distributed over $\{0,1\}^n$. Fix $n$ for which this happens. Then we rewrite these distributions more explicitly:

$$\begin{aligned}
&\{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(U_0^n,U_1^n)),\mathcal{A}(1^n)],U_{1-b}^n)\}\\
&\qquad=\{s_0\leftarrow U_0^n; s_1\leftarrow U_1^n, z\leftarrow \bar{U}^n; v\leftarrow \mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(s_0,s_1)),\mathcal{A}(1^n)]:\Gamma(v,s_{1-b})\}\\
&\{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(U_0^n,U_1^n)),\mathcal{A}(1^n)],\bar{U}^n)\}\\
&\qquad=\{s_0\leftarrow U_0^n; s_1\leftarrow U_1^n, z\leftarrow \bar{U}^n; v\leftarrow \mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(s_0,s_1)),\mathcal{A}(1^n)]:\Gamma(v,z)\}
\end{aligned}$$

Next, we use $\mathcal{A}$ to construct $\mathcal{A}'$ that breaks the defensible privacy of $\pi_{\mathrm{OT}}$ with respect to a corrupted receiver. We use the experiment formulation of defensible privacy for the bit OT protocol. Towards this, we consider the following sequence of distributions:

$$\begin{aligned}
\overline{\mathsf{hyb}}_i(n)=\{s_0\leftarrow U_0^n; s_1\leftarrow U_1^n, z\leftarrow \bar{U}^n; v\leftarrow \mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(s_0,s_1)),\mathcal{A}(1^n)]:\\
\Gamma(v,(z^1,\ldots,z^i,s_{1-b}^{i+1},\ldots,s_{1-b}^n))\}
\end{aligned}$$

where $z=(z^1,\ldots,z^n)$ and $s_0=(s_0^1,\ldots,s_0^n)$ and $s_1=(s_1^1,\ldots,s_1^n)$. Observe that

$$\begin{aligned}
\overline{\mathsf{hyb}}_0(n) &= \{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(U_0^n,U_1^n)),\mathcal{A}(1^n)],U_{1-b}^n)\}\\
\overline{\mathsf{hyb}}_n(n) &= \{\Gamma(\mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n,(U_0^n,U_1^n)),\mathcal{A}(1^n)],\bar{U}^n)\}.
\end{aligned}$$

Then using a standard hybrid argument we can conclude that there exists an index $i$ such that $D$ distinguishes $\mathsf{hyb}_{i-1}(n)$ and $\mathsf{hyb}_i(n)$ with probability at least $\frac{1}{np(n)}$. More precisely,

$$\left|\Pr[D(\overline{\mathsf{hyb}}_i(n))=1]-\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n))=1]\right|>\frac{1}{p(n)}.$$

Without loss of generality, we assume that

$$\Pr[D(\overline{\mathsf{hyb}}_i(n))=1]-\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n))=1]>\frac{1}{p(n)}. \tag{9}$$

Now, consider a machine $\mathcal{A}'$ that is interacting externally with a sender on input $(s_0,s_1)$ in the protocol $\pi_{\mathrm{OT}}$. $\mathcal{A}'$ internally incorporates $\mathcal{A}$ and proceeds as follows. It starts by emulating an execution with $\mathcal{A}$ by supplying the sender's messages in $\pi_{\mathrm{OT}}^n$ which implies $n$ parallel OT executions. Specifically, $\mathcal{A}'$ supplies random inputs for the sender in all but the $i^{th}$ execution, for which it forwards externally to the sender that participates in $\pi_{\mathrm{OT}}$. Denote by $(s_0^j,s_1^j)$ the sender's selected inputs for every $j\neq i$. Upon completion, $\mathcal{A}'$ receives a defense from $\mathcal{A}$. If the defense is not good $\mathcal{A}'$ aborts. Else, $\mathcal{A}'$ computes $w=(w_1,\ldots,w_n)$ as follows:

- $w_j=z^j$ where $z^j$ is sampled at random from $\{0,1\}$ for $j<=i$.
- $w_j=s_{u_j}^j$ for $j>i$, where $u_j$ is the receiver's input in the $j^{th}$ execution which can be obtained from the defense output by $\mathcal{A}$.

Next, $\mathcal{A}'$ invokes $D$ on input $(v, w)$ where $v$ is $\mathcal{A}$'s internally generated view. Let $b$ be the output of $D$ on these inputs and $(u_i, \rho_R)$ be the defense of $\mathcal{A}$ in the $i^{th}$ interaction. Then $\mathcal{A}'$ outputs a defense $(u_i, \rho_R)$ and defined its guess for the external sender's other input by $b \oplus w_i$. By construction, we have that $(v, w)$ are sampled in the internal emulation according to $\overline{\mathsf{hyb}}_i(n)$. This means that $\mathcal{A}'$ succeeds in the experiment $\mathsf{Expt}_{\pi_{\mathrm{OT}}}$ when it is given

$$D(\overline{\mathsf{hyb}}_i(n)) \oplus w_i = s_{1-u_i}.$$

Observe that if $w_i = s_{1-u_i}$ then $(v, w)$ in the internal emulation of $\mathcal{A}'$ is distributed according to $\overline{\mathsf{hyb}}_{i-1}(n)$. This means that:

$$\Pr[\mathcal{A}' \text{ wins } \mathsf{Expt}_{\pi_{\mathrm{OT}}} | w_i = s_{1-u_i}] = \Pr[D(\overline{\mathsf{hyb}}_{i-1}(n) = 0]. \tag{10}$$

Next, we introduce a new distribution $\widehat{U}_j^n$ that is identical to $\widetilde{U}_j^n$ with the exception that the $(j+1)^{st}$ bit in $\bar{U}_j^n$ is flipped. More precisely, for $j \in [n]$,

$$\widehat{\mathsf{hyb}}_j(n) = \{s_0 \leftarrow U_0^n; s_1 \leftarrow U_1^n, z \leftarrow \bar{U}^n; v \leftarrow \mathbf{View}_{\mathcal{A}}[\mathrm{Sen}(1^n, (s_0, s_1)), \mathcal{A}(1^n)] : \Gamma(v, (z^1, \ldots, z^j, 1 \oplus s_{1-b}^{j+1}, \ldots, s_{1-b}^n))\}$$

where $z = (z^1, \ldots, z^n)$, $s_0 = (s_0^1, \ldots, s_0^n)$ and $s_1 = (s_1^1, \ldots, s_1^n)$. Now, since the bits in the $i^{th}$ position are complement of each other in $\widehat{U}_{i-1}^n$ and $\widetilde{U}_{i-1}^n$, and the $i^{th}$ bit is randomly distributed in $\widetilde{U}_i^n$ we have that

$$\Pr[D(\overline{\mathsf{hyb}}_i(n)) = 1] = \frac{1}{2}\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n)) = 1] + \frac{1}{2}\Pr[D(\widehat{\mathsf{hyb}}_{i-1}(n)) = 1]. \tag{11}$$

Moreover, $\mathcal{A}'$ succeeds if $D(\widehat{\mathsf{hyb}}_{i-1}(n)) = 1$ when $w_i \neq s_{1-u_i}$. More precisely

$$\Pr[\mathcal{A}' \text{ wins } \mathsf{Expt}_{\pi_{\mathrm{OT}}} | w_i \neq s_{1-u_i}] = \Pr[D(\widehat{\mathsf{hyb}}_i(n) = 1]. \tag{12}$$

Now, since both $s_{1-r}$ and $w_i$ are chosen at random we have that:

$$\begin{aligned}
&\Pr[\mathcal{A}' \text{ wins } \mathsf{Expt}_{\pi_{\mathrm{OT}}}] \\
&\quad = \Pr[\mathcal{A}' \text{ wins } \mathsf{Expt}_{\pi_{\mathrm{OT}}} | w_i \neq s_{1-u_i}] \Pr[w_i \neq s_{1-u_i}] \\
&\qquad + \Pr[\mathcal{A}' \text{ wins } \mathsf{Expt}_{\pi_{\mathrm{OT}}} | w_i = s_{1-r}] \Pr[w_i = s_{1-u_i}] \\
&\quad = \frac{1}{2}\Pr[D(\widehat{\mathsf{hyb}}_{i-1}(n)) = 1] + \frac{1}{2}\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n)) = 0] && \text{(Using Equations 12 and 10)} \\
&\quad = \left(\Pr[D(\mathsf{hyb}_i(n)) = 1] - \frac{1}{2}\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n)) = 1]\right) \\
&\qquad\qquad + \frac{1}{2}\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n)) = 0] && \text{(Using Equation 11)} \\
&\quad = \left(\Pr[D(\mathsf{hyb}_i(n)) = 1] - \frac{1}{2}\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n)) = 1]\right) + \left(\frac{1}{2} - \frac{1}{2}\Pr[D(\overline{\mathsf{hyb}}_{i-1}(n)) = 1]\right) \\
&\quad = \frac{1}{2} + \left(\Pr[D(\overline{\mathsf{hyb}}_i(n)) = 1] - \Pr[D(\overline{\mathsf{hyb}}_{i-1}(n)) = 1]\right) \\
&\quad \geq \frac{1}{2} + \frac{1}{p(n)} && \text{(Using Equation 9).}
\end{aligned}$$

This contradicts the defensible privacy against a corrupted receiver in protocol $\pi_{\mathrm{OT}}$. ■

# C Extractable Commitments from PKE in the CRS model

In this section, we realize the $\mathcal{F}_{\text{EXTCOM}}$ functionality in the $\mathcal{F}_{\text{CRS}}$ model. We recall the formal definition of $\mathcal{F}_{\text{CRS}}$ in Figure 5 and note that this functionality generates a CRS for any pair of parties. At a high-level, we obtain an extractable commitment using a IND-CPA PKE. Loosely speaking, the common reference string contains a public-key that is sampled using the key-generation algorithm. Moreover, the trapdoor for the CRS is the corresponding secret-key. In the real world, no adversary knows that secret-key and hence it does not know the corresponding CRS trapdoor. In order to implement extractable commitments, our protocol requires from the commitment sender to simply encrypt its message $m$ using the public-key that is placed in the CRS. Decommitment is carried out by asking the sender to provide the randomness used to encrypt $m$.

**Functionality $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$**

$\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$ runs with parties $P_1, \ldots, P_n$ and adversary $\mathcal{S}$. This functionality is parameterized by an algorithm $\mathcal{D}$.

- On input $(\text{CRS}, sid, P_i)$ from $P_i$, it runs the sampling algorithm $\mathcal{D}$ with uniformly distributed random input $\rho \in \{0,1\}^n$ to obtain a reference string $R = \mathcal{D}(\rho)$. Store $(P_i, \rho, R)$ and send $(sid, P_i, R)$ to $P_i$ and adversary $\mathcal{S}$. Next, when receiving $(sid, P_j)$ from $P_j$, send $(sid, P_j, R)$ to $P_j$ and $\mathcal{S}$, and halt.

Figure 5: The common reference string functionality.

**Protocol $\Pi_{\text{EXTCOM}}$**

- **CRS:** The string consists of a public-key PK obtained by running the key-generation algorithm $\mathsf{Gen}(1^n)$ on a uniformly sampled random string.
- **Commit phase:**
  1. On input $(\mathsf{commit}, sid, ssid, P_i, P_j, m)$ where $m \in \{0,1\}^n$, party $P_i$ computes $c = \mathsf{Enc}_{\mathsf{PK}}(m; r)$ for a uniformly distributed string $r$. $P_i$ records $(sid, ssid, P_j, r, m)$, and sends $s = (sid, ssid, P_i, c)$ to $P_j$.
  2. $P_j$ receives and records $s$, and outputs $(\mathsf{receipt}, sid, ssid, P_i, P_j)$. $P_j$ ignores any later commit messages from $P_i$ with the same $(sid, ssid)$.
- **Reveal phase:**
  1. On input $(\mathsf{decommit}, sid, ssid)$, party $P_i$ retrieves $(sid, ssid, P_j, (m, r))$ and sends $(sid, ssid, (m, r))$ to $P_j$.
  2. Upon receiving $(sid, ssid, (m, r))$ from $P_i$, $P_j$ checks that it has a tuple $(sid, ssid, P_i, c)$. If yes, then it verifies that $c = \mathsf{Enc}_{\mathsf{PK}}(m; r)$. Otherwise, it ignores the message.

Figure 6: Protocol $\Pi$ to realize $\mathcal{F}_{\text{EXTCOM}}$.

**Lemma C.1.** *Let $\Pi'$ be a protocol in the $\mathcal{F}_{\text{EXTCOM}}$-hybrid model and assume the existence of IND-CPA PKE. Then there exist a protocol $\Pi$ in the $\mathcal{F}_{\text{CRS}}$-hybrid model such that for every* PPT *adversary $\mathcal{A}$ there exists a* PPT *simulator $\mathcal{S}$, where for every* PPT *environment $\mathcal{Z}$ the following two ensembles are indistinguishable:*

- $\mathbf{View}^{\mathcal{F}_{\text{CRS}}}_{\Pi,\mathcal{A},\mathcal{Z}}(n)$
- $\mathbf{View}^{\mathcal{F}_{\text{EXTCOM}}}_{\Pi',\mathcal{S},\mathcal{Z}}(n)$.

**Proof:** Given any protocol $\Pi'$ in the $\mathcal{F}_{\text{EXTCOM}}$-hybrid model, then protocol $\Pi$ in the real model is constructed from $\Pi'$ by instantiating the $\mathcal{F}_{\text{EXTCOM}}$ functionality using $\Pi_{\text{EXTCOM}}$ (where $\mathcal{F}_{\text{EXTCOM}}$ is defined in Section 2.6). More precisely, $\Pi$ is obtained from $\Pi'$ as follows:

- $P_i$ *sends* $(\mathsf{init}-\mathsf{commit}, sid, P_i, P_j)$ *to* $\mathcal{F}_{\text{EXTCOM}}$ *in* $\Pi'$. Instead, $P_i$ continues as follows in $\Pi$. It retrieves $(sid, P_i, \text{crs})$ if such tuple already exists. Otherwise, it sends $(\mathsf{CRS}, sid, P_i)$ to $\mathcal{F}_{\text{CRS}}$ and obtains $(sid, P_i, \text{crs})$ from the $\mathcal{F}_{\text{CRS}}$.
- $P_i$ *receives* $(\mathsf{public}-\mathsf{params}, sid, P_i, pp)$ *from* $\mathcal{F}_{\text{EXTCOM}}$ *in* $\Pi'$. $P_i$ ignores this message since it never receives such message in the $\mathcal{F}_{\text{CRS}}$-hybrid model.
- $P_i$ *sends* $(\mathsf{commit}, sid, ssid, P_i, P_j, m)$ *to* $\mathcal{F}_{\text{EXTCOM}}$ *in* $\Pi'$. Instead, $P_i$ retrieves the tuple $(sid, P_i, \text{crs})$ and interacts with $P_j$ by executing $\Pi_{\text{EXTCOM}}$ and playing the role of the sender with common input $(sid, \text{crs})$ and private input $m$. If no tuple of the form $(sid, P_i, \text{crs})$ exists, $P_i$ aborts.
- $P_j$ *receives* $(\mathsf{receipt}, sid, ssid, P_i, P_j, \mathsf{trans})$ *from* $\mathcal{F}_{\text{EXTCOM}}$ *in* $\Pi'$. Instead, $P_j$ retrieves the tuple $(sid, P_i, \text{crs})$ and interacts with $P_i$ by executing $\Pi_{\text{EXTCOM}}$ and playing the role of the receiver with parameters $(sid, \text{crs})$. If no such tuple $(sid, P_i, \text{crs})$ exists, $P_j$ ignores the communication.
- $P_i$ *sends* $(\mathsf{decommit}, sid, ssid, P_i, P_j)$ *to* $\mathcal{F}_{\text{EXTCOM}}$ *in* $\Pi'$. In this case, $P_i$ sends $(sid, ssid, r, m)$ to $P_j$. Recall that this information is stored by $P_i$ at the end of the commit phase.

We now prove that $\Pi_{\text{EXTCOM}}$ induces algorithm pp and two protocols $\Pi_L$ and $\Pi_R$, where $\Pi_L$ implies straight-line extractability and $\Pi_R$ is a statistically-binding commitment scheme in the plain model with no access to $\mathcal{F}_{\text{CRS}}$.

- pp is the key-generation algorithm for the public-key encryption scheme Gen that returns $(\text{PK}, \text{SK})$ on input $1^n$.
- $\Pi_L$ is the protocol, where in the commit phase the sender on input $m$, samples a random string $r$ of appropriate length and computes $c = \mathsf{Enc}_{\mathsf{PK}}(m; r)$ where the common input for the protocol is $(sid, \text{PK})$. It then forwards $(sid, ssid, \mathsf{trans})$ where $\mathsf{trans} = c$ to the receiver. Following the transmission, the receiver outputs $z = m$, where $m = \mathsf{Dec}_{\mathsf{SK}}(c)$ and the sender outputs $\gamma = (m, r)$.
- $\Pi_R$ is the protocol where the sender on input $m$ samples a random string $r$ of appropriate length and computes $c = \mathsf{Enc}_{\mathsf{PK}}(m; r)$ where the common input for the protocol is $(sid, \text{PK})$. Then it forwards $(sid, ssid, \mathsf{trans})$ where $\mathsf{trans} = c$ to the receiver where $c = \mathsf{Enc}_{\mathsf{PK}}(m; r)$ and $m$ is the sender's input.

Furthermore, the function opening returns $m$ when given input $(\mathsf{trans}, \gamma)$, if $c = \mathsf{Enc}_{\mathsf{PK}}(m; r)$ where $\mathsf{trans} = c$ and $\gamma = (m, r)$. Next, given an adversary $\mathcal{A}$ that interacts with the parties running protocol $\Pi$ in the $\mathcal{F}_{\text{CRS}}$-hybrid model, we construct an adversary $\mathcal{S}$ with access to $\mathcal{F}_{\text{EXTCOM}}$ which simulates the execution of protocol $\Pi$ with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal execution of $\Pi'$ with $\mathcal{S}$ and $\mathcal{F}_{\text{EXTCOM}}$ from a real execution of $\Pi$ with $\mathcal{A}$. Recall that $\mathcal{S}$ interacts with the ideal functionality $\mathcal{F}_{\text{EXTCOM}}$ and with the environment $\mathcal{Z}$. $\mathcal{S}$ begins by internally emulating $\mathcal{A}$ and running a simulated interaction with $\mathcal{A}$, environment $\mathcal{Z}$ and the parties that run the protocol. We will focus on the interaction between the parties $P_i$ and $P_j$ for arbitrary $i$ and $j$, where $P_i$ is designated as the sender and $P_j$ is designated as the receiver. At a high-level, $\mathcal{S}$ essentially sets up the CRS in the internal emulation of $\mathcal{A}$ using the public-parameters obtained from $\mathcal{F}_{\text{EXTCOM}}$ and for the remaining just forwards the communication of $\mathcal{A}$ directly to $\mathcal{F}_{\text{EXTCOM}}$. $\mathcal{S}$ proceeds as follows:

- **Simulating the communication with $\mathcal{Z}$:** Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is feeded internally to $\mathcal{A}$. Similarly all communications from $\mathcal{A}$ intended for the environment are forwarded back to $\mathcal{Z}$.

- $P_i$ **is honest and** $P_j$ **is honest:** The actions of the honest parties are determined by functionality $\mathcal{F}_{\text{EXTCOM}}$.

- $P_i$ **is corrupted and** $P_j$ **is honest:**

  *Initialization:* Upon receiving $(\mathsf{init}-\mathsf{commit}, sid, ssid, P_i, P_j)$ from $\mathcal{A}$ (on behalf of $P_i$), $\mathcal{S}$ checks if it has the tuple $(sid, P_i, \text{PK})$. Otherwise, it forwards $(\mathsf{init}-\mathsf{commit}, sid, ssid, P_i, P_j)$ to $\mathcal{F}_{\text{EXTCOM}}$ and receives $(\mathsf{public}-\mathsf{params}, sid, P_i, \text{PK})$ from $\mathcal{F}_{\text{EXTCOM}}$. Next, $\mathcal{S}$ simulates the $\mathcal{F}_{\text{CRS}}$-functionality and sets the reference string to PK. More precisely, it sends $(sid, P_i, \text{PK})$ to $\mathcal{A}$ and locally stores $(sid, P_i, \text{PK})$.

  *Commit phase:* $\mathcal{S}$ interacts with the corrupted sender $\mathcal{A}$ within $\Pi_{\text{EXTCOM}}$, playing the role of the receiver. More precisely, it receives $(sid, ssid, c)$ from $\mathcal{A}$ and checks whether it previously stored $(sid, P_i, \text{PK})$, and ignores this message from $\mathcal{A}$ if it cannot find such a message. Otherwise, it sends $(sid, ssid, c)$ to $\mathcal{F}_{\text{EXTCOM}}$.

  *Decommit Phase:* Upon receiving $(sid, ssid, r, m)$ from $\mathcal{A}$, $\mathcal{S}$ sends $(\mathsf{decommit}, sid, ssid, (m, r))$ to $\mathcal{F}_{\text{EXTCOM}}$.

- $P_i$ **is honest and** $P_j$ **is corrupted:**

  *Initialization:* Upon receiving $(\mathsf{public}-\mathsf{params}, sid, P_i, \text{PK})$ from $\mathcal{F}_{\text{EXTCOM}}$, $\mathcal{S}$ simulates the $\mathcal{F}_{\text{CRS}}$-functionality and sets the reference string to PK. More precisely, it sends $(sid, P_i, \text{PK})$ to $\mathcal{A}$ and locally stores $(sid, P_i, \text{PK})$.

  *Commit phase:* $\mathcal{S}$ interacts with $\mathcal{A}$ using $\Pi_{\text{EXTCOM}}$, playing the role of the sender. More precisely, it receives $(sid, ssid, (\text{PK}, c))$ from $\mathcal{F}_{\text{EXTCOM}}$ and feeds $(sid, ssid, P_i, (\text{PK}, c))$ to $\mathcal{A}$.

  *Decommit phase:* $\mathcal{S}$ receives $(\mathsf{decommit}, sid, ssid, P_i, (m, r))$ from $\mathcal{F}_{\text{EXTCOM}}$. Internally, $\mathcal{S}$ feeds $(sid, ssid, (m, r))$ to $\mathcal{A}$.

Before we prove the correctness of the simulation we need to establish that $(\mathsf{pp}, \Pi_L, \Pi_R)$ are as in Definition 2.9. First, we observe that if $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is IND-CPA PKE with perfect decryption, then $\Pi_L$ and $\Pi_R$ are perfectly binding and $\Pi_R$ is computationally hiding. The only thing remaining to show is that $\Pi_L$ is extractable. More precisely, we need to show that for $(\mathsf{trans}, \gamma)$ and $z$, that are determined at the end of the commit phase, the probability that $\mathsf{opening}(\mathsf{trans}, \gamma) \neq \perp$ and $\mathsf{opening}(\mathsf{trans}, \gamma) \neq z$ is negligible. Recall that in our construction, $\mathsf{trans} = c$, $z = m$ and $\gamma = (m, r)$. Now, since we rely on a public-key encryption scheme with 0-decryption error, for a given any ciphertext $c$, there exists only one $m$ for which there exists $r$ such that $c = \mathsf{Enc}_{\mathsf{PK}}(m, r)$. Moreover, $m = \mathsf{Dec}_{\mathsf{SK}}(c)$. Hence, this property holds.

We now prove that $\mathcal{Z}$ cannot distinguish an interaction of $\Pi$ with $\mathcal{A}$ in the $\mathcal{F}_{\text{CRS}}$-hybrid model from an interaction of $\mathcal{A}'$ in the $\mathcal{F}_{\text{EXTCOM}}$-hybrid model with $\mathcal{S}$. Observe that internal emulation of $\mathcal{S}$ with $\mathcal{A}$ proceeds identically to the real execution. This is because the CRS in the internal emulation is distributed identically to the real execution and $\mathcal{F}_{\text{EXTCOM}}$ follows the code of the honest sender and honest receiver in $\Pi_{\text{EXTCOM}}$ when interacting with a corrupted receiver and corrupted sender respectively. □