On the Complexity of UC Commitments*

Juan A. Garay† Yuval Ishai‡ § Ranjit Kumaresan‡ Hoeteck Wee¶ ||

May 14, 2014

Abstract

Motivated by applications to secure multiparty computation, we study the complexity of realizing universally composable (UC) commitments. Several recent works obtain practical UC commitment protocols in the common reference string (CRS) model under the DDH assumption. These protocols have two main disadvantages. First, even when applied to long messages, they can only achieve a small constant rate (namely, the communication complexity is larger than the length of the message by a large constant factor). Second, they require computationally expensive public-key operations for each block of each message being committed.

Our main positive result is a UC commitment protocol that simultaneously avoids both of these limitations. It achieves an optimal rate of 1 (strictly speaking, $1 - o(1)$) by making only few calls to an ideal oblivious transfer (OT) oracle and additionally making a black-box use of a (computationally inexpensive) PRG. By plugging in known efficient protocols for UC-secure OT, we get rate-1, computationally efficient UC commitment protocols under a variety of setup assumptions (including the CRS model) and under a variety of standard cryptographic assumptions (including DDH). We are not aware of any previous UC commitment protocols that achieve an optimal asymptotic rate.

A corollary of our technique is a rate-1 construction for *UC commitment length extension*, that is, a UC commitment protocol for a long message using a single ideal commitment for a short message. The extension protocol additionally requires the use of a semi-honest (stand-alone) OT protocol. This raises a natural question: can we achieve UC commitment length extension while using only inexpensive PRG operations as is the case for stand-alone commitments and UC OT? We answer this question in the negative, showing that the existence of a semi-honest OT protocol is necessary (and sufficient) for UC commitment length extension. This shows, quite surprisingly, that UC commitments are qualitatively different from both stand-alone commitments and UC OT.

Keywords: Universal composability, UC commitments, oblivious transfer.

1 Introduction

A *commitment scheme* is a digital analogue of a locked box. It enables one party, called the *committer*, to transfer a value to another party, called the *receiver*, while keeping it hidden, and later reveal it while guar-

*Research received funding from the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC.

†Yahoo Labs. Email: garay@yahoo-inc.com

‡Department of Computer Science, Technion. Email: {yuvali, ranjit}@cs.technion.ac.il

§Supported in part by ISF grant 1361/10 and BSF grant 2012378.

¶ENS, Paris, France. Email: wee@di.ens.fr

||CNRS (UMR 8548) and INRIA. Part of this work was done at George Washington University, supported by NSF Award CNS-1237429.

anteeing to the receiver its originality. Commitment schemes are a fundamental building block for cryptographic protocols withstanding active adversarial attacks. As such, efficient implementations of the latter—particularly in realistic complex environments where they are to execute—crucially hinge on them. Such complex environments are today epitomized by the universal composability (UC) framework [6], which allows for a protocol to run concurrently and asynchronously with arbitrarily many others, while guaranteeing its security.

The first constructions of UC commitments were given by Canetti *et al.* [7, 8] as a feasibility result. (It was also shown in [7] that it is impossible to construct UC commitments in the plain model, and that some setup such as a *common reference string* (CRS) is required.) Since then, and motivated by the above, a series of improvements (e.g., [14, 13, 28, 19, 33, 4, 1, 25]) culminated in constructions achieving under various cryptographic assumptions *constant* communication rate and practical computational complexity, making it possible to commit to, say, L group elements by sending $O(L)$ group elements and performing $O(L)$ public-key operations (e.g., exponentiations).

Shortcomings—as well as ample room for improvement, however, remain, as the constant rate currently achieved is small and the computational cost per committed bit is high. This is the case even when committing to long messages and even when ignoring the cost of offline interaction that does not depend on the committed message. More concretely, the communication complexity is bigger than the length of the message by a large constant factor, and the online computation includes a large number of computationally expensive public-key operations for each block of the message being committed.[1] This is not satisfactory when considering concrete applications where UC commitments are used, such as UC secure computation and UC zero-knowledge. (See [4, 28, 1] for additional motivation on these applications.)

1.1 Our results

We obtain both positive and negative results on the complexity of UC commitments. Our main positive result is a UC commitment protocol which simultaneously overcomes both of these limitations. Specifically, it achieves an *optimal rate* of 1 (strictly speaking, $1 - o(1)$) by making only few calls to an ideal oblivious transfer (OT) oracle and additionally making a black-box use of a (computationally inexpensive) PRG. By plugging in known efficient protocols for UC-secure OT (e.g., [34]), we get rate-1, computationally efficient UC commitment protocols under a variety of setup assumptions (including the CRS model) and under a variety of standard cryptographic assumptions (including DDH). We are not aware of any previous UC commitment protocols which achieve an optimal asymptotic rate.

Basic construction. Our main idea is to use a simple code-based generalization of the standard construction of commitment from δ-Rabin-string-OTs [11, 26, 24, 18]. The key observation is that the use of a rate-1 encoding scheme with a judicious choice of parameters yields a rate-1 construction of UC commitments. In a bit more detail, we will identity the message space $\{0,1\}^\ell$ with $\mathbb{F}^n$, and encode the message as a codeword in $\mathbb{F}^{n(1+\delta)}$ using the multi-secret sharing scheme in [17]. The codeword is then transmitted to the receiver using $n(1+\delta)$ calls to a δ-Rabin-OT functionality, which can be in turn realized from a standard OT functionality. The sender would send in total roughly

$$\ell(1+\delta) + n(1+\delta) \cdot 1/\delta \cdot \kappa$$

bits in the commit phase, where the first term comes from the encoding and the second term is the additive overhead incurred by simulating a δ-Rabin-OT functionality using a standard OT functionality and a PRG

[1] Recent constructions [28, 4] that work over standard DDH groups require at least 10 group elements and at least 20 public key operations per commitment instance. A very recent work by [25] (improving over [16]) requires 5 group elements in a bilinear group (assuming SXDH).

with security parameter κ. Setting $n = 1/\delta = \ell^{1/3}$ yields a commitment scheme where the sender sends $\ell + o(\ell)$ bits and performs $o(\ell)$ public-key operations for a single commitment to a ℓ-bit string.

Efficiency improvements. Next, we show how to further reduce the computational complexity of the basic construction by using OT extension [2, 23, 24]. Our improvement ideally suits the setting where we need to perform a large number of commitments in a single parallel commit phase (with potentially several reveal phases), as with applications involving cut-and-choose. In particular, we show that the number of calls to the OT oracle can be made independent of the number of instances of UC commitments required. (Note that such a result does not follow from multiple applications of the basic construction.) We stress that when handling a large number of commitment instances (say, in garbled circuit applications of cut-and-choose), the number of public-key operations plays a significant role (perhaps more than the communication) in determining efficiency. While current state-of-the-art UC commitment protocols [28, 4] suffer from the need to perform many computationally expensive public-key operations, our result above enables us to obtain better computational as well as overall efficiency.

UC commitment length extension. Another corollary of our technique is a rate-1 construction for *UC commitment length extension*, that is, a UC commitment protocol for a long message using a single ideal commitment for a short message. The extension protocol additionally requires the use of a semi-honest (stand-alone) OT protocol. This raises a natural question of whether we can achieve UC commitment length extension while using only inexpensive PRG operations as is the case for stand-alone commitments and UC OT. We answer this question in the negative, showing that the existence of a semi-honest OT protocol is necessary (and sufficient) for UC commitment length extension. This shows that UC commitments are qualitatively different from both stand-alone commitments and UC OT, which can be extended using any PRG [2], and are similar to adaptively-secure OT whose extension requires the existence of (non-adaptively secure) oblivious transfer [29].

Open questions. We note that our constructions are only secure against a static (non-adaptive) adversary; we leave the extension to adaptive security for future work. Another open question is to obtain a UC commitment scheme where the sender sends $\ell + \mathrm{poly}(\kappa)$ bits to commit to a ℓ-bit string, as is the case for both stand-alone commitment and UC OT.

1.2 Related work

We already mentioned above the series of results leading to constant-rate UC-commitments. Here we give a brief overview. Canetti *et al.* [7, 8] were the first to construct (inefficient) UC commitments in the CRS model from general assumptions, and also achieve adaptive security. Shortly thereafter, Damgård and Nielsen [14] presented UC commitments with $O(1)$ exponentiations for committing to a single group element. Their construction is based on N-residuosity and p-subgroup assumptions, and is also adaptively secure (without erasures), but requires a CRS that grows linearly with the number of parties. A construction of Damgård and Groth [13], also adaptively secure without erasures and based on the strong RSA assumption, requires a fixed-length CRS.

An important improvement in concrete efficiency was presented recently by Lindell [28]; this is achieved for static corruptions based on the DDH assumption in the CRS model. Blazy *et al.* [4] build on Lindell's scheme to achieve adaptive security (assuming erasures); they also obtain improvements in concrete efficiency. Fischlin *et al.* [16] also build on Lindell's scheme and present a non-interactive scheme using Groth-Sahai proofs [21]. Furthermore, they also provide an adaptively secure variant (with erasures) based

on the DLIN assumption on symmetric bilinear groups. As mentioned above, none of these works achieve rate 1. We provide a concrete analysis of our protocol, with a comparison to [28, 4] in Section 3.3.

A code-based construction of UC commitments from OT was recently used by Frederiksen *et al*. [18] as part of an efficient protocol for secure two-party computation. While this construction uses a similar high level technique as our basic construction, its suggested instantiation in [18] only achieves a small constant rate.

Our work also considers the extension of UC commitments. We mainly focus on the goal of *length extension*, namely using an ideal commitment to a short string for implementing a UC commitment to a long string. For standalone commitments, such a length extension is easy to implement using any PRG. This is done similarly to the standard use of a PRG for implementing a hybrid encryption scheme. It was previously shown by Kraschewski [27] that this simple extension technique does not apply to UC commitments. We strengthen this negative result to show that *any* extension protocol for UC commitments implies oblivious transfer. Similar negative results for adaptively secure OT extension were obtained by Lindell and Zarosim [29], and for reductions between finite functionalities by Maji *et al*. [30]. Negative results for statistical UC coin-tossing extension were obtained by Hofheinz *et al*. [22].

In an independent work, Damgård *et al*. [12] also construct UC commitments using OT, PRG, and secret sharing as the main ingredients. While the basic approach is closely related to ours, the concrete constructions are somewhat different, leading to incomparable results. In particular, a major goal in [12] is to optimize the asymptotic computational complexity as a function of the security parameter, achieving in one variant constant (amortized) computational overhead for the receiver. Moreover, they achieve both additive and multiplicative properties for UC commitments, which are not considered in our work.

Organization of the paper. The rest of the paper is organized as follows. Model, definitions, and basic functionalities are presented in Section 2. Our main construction—rate-1 UC commitment from OT—is presented in Section 3, together with the case of multiple commitment instances and a concrete efficiency analysis. Finally, the treatment of UC commitment extension—rate-1 construction and necessity of OT—is presented in Section 4. Due to space limitations, only proof sketches are presented in the main body; full proofs as well as complementary material can be found in the appendix.

2 Model and Definitions

In this section we introduce some notation and definitions that will be used throughout the paper. We denote the computational security parameter by κ, and the statistical security parameter by σ. A function μ is negligible if for every polynomial p there exists an integer N such that for every $n > N$ it holds that $\mu(n) < 1/p(n)$.

In this paper we will be concerned with efficient universally composable (UC) [6] realizations of functionalities such as commitments. We include a succint description of the UC basics in Appendix A. Assuming already some familiarity with the framework, we note that it is possible to consider variants of the definition of UC security in which the order of quantifiers is "$\forall\mathcal{A}\exists\mathcal{S}\forall\mathcal{Z}$". Contrast this with our definition (and also the definition in [28]) in which the order of quantifiers is "$\exists\mathcal{S}\forall\mathcal{Z}\forall\mathcal{A}$". Both definitions are equivalent as long as $\mathcal{S}$, in the former definition makes only a blackbox use of $\mathcal{A}$ [6]. Indeed, this will be the case in our constructions. Therefore, as in [28], we demonstrate a single simulator $\mathcal{S}$ that works for all adversaries and environments, and makes only a blackbox use of the adversary. (In this case, one may also denote the ideal process by $\text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}$.)

We will sometimes explicitly describe the functionalities we realize. For instance, if a functionality $\mathcal{F}$ accepts inputs only of a certain length ℓ, then we will use the notation $\mathcal{F}[\ell]$ to denote this functionality. We let $\mathsf{cc}(\mathcal{F})$ denote the communication cost, measured in bits, of realizing $\mathcal{F}$ in the *plain model*.

Functionality $\mathcal{F}_{\text{MCOM}}$

$\mathcal{F}_{\text{MCOM}}$ with session identifier sid proceeds as follows, running with parties $P_1, \ldots, P_n$, a parameter 1^κ, and an adversary $\mathcal{S}$:

- Commit phase: Upon receiving a message $(\mathsf{commit}, sid, ssid, s, r, m)$ from P_s where $m \in \{0,1\}^\ell$, record the tuple $(ssid, s, r, m)$ and send the message $(\mathsf{receipt}, sid, ssid, s, r)$ to P_r and $\mathcal{S}$. (The length of the strings ℓ is fixed and known to all parties.) Ignore any future commit messages with the same $ssid$ from P_s to P_r.
- Decommit phase: Upon receiving a message $(\mathsf{reveal}, sid, ssid)$ from P_s: If a tuple $(ssid, s, r, m)$ was previously recorded, then send the message $(\mathsf{reveal}, sid, ssid, s, r, m)$ to P_r and $\mathcal{S}$. Otherwise, ignore.

Figure 1: Functionality $\mathcal{F}_{\text{MCOM}}$ for multiple commitments.

Functionality $\mathcal{F}_{\text{OT}}^N$

$\mathcal{F}_{\text{OT}}^N$ with session identifier sid proceeds as follows, running with parties $P_1, \ldots, P_n$, a parameter 1^κ, and an adversary $\mathcal{S}$:

- Upon receiving a message $(\mathsf{sender}, sid, ssid, s, r, x_1, \ldots, x_N)$ from P_i, where each $x_j \in \{0,1\}^\ell$, record the tuple $(sid, ssid, s, r, x_1, \ldots, x_N)$. (The length of the strings ℓ is fixed and known to all parties.) Ignore any future sender messages with the same $sid, ssid$ pair from P_s to P_r.
- Upon receiving a message $(\mathsf{receiver}, sid, ssid, s, r, q)$ from P_r, where $q \in [N]$, send $(sid, ssid, s, r, x_q)$ to P_r and $(sid, ssid, s, r)$ to P_s, and halt. (If no $(\mathsf{sender}, sid, ssid, s, r, \ldots)$ message was previously sent, then send nothing to P_r.)

Figure 2: Functionality $\mathcal{F}_{\text{OT}}^N$ for 1-out-of-N oblivious transfer. We omit superscript N when $N = 2$.

The multi-commitment ideal functionality $\mathcal{F}_{\text{MCOM}}$, which is the functionality that we UC realize in this work, is given in Figure 1. As mentioned above, $\mathcal{F}_{\text{MCOM}}[\ell]$ will explicitly denote that the functionality accepts inputs of length exactly ℓ. We will be giving our constructions in the OT-hybrid model. The oblivious transfer functionality $\mathcal{F}_{\text{OT}}^N$, capturing 1-out-of-$N$ OT for $N \in \mathbb{Z}$, is described in Figure 2. When $N = 2$, this is the standard 1-out-of-2 string-OT functionality, denoted by $\mathcal{F}_{\text{OT}}$. The δ-Rabin-string-OT functionality, denoted $\mathcal{F}_{\text{OT}_{\text{R}}}^\delta$, is described in Figure 3.

3 Rate-1 UC Commitments from OT

A recent line of work has focused on the practical efficiency of UC commitment in the CRS model [28, 4, 1, 19, 33]. In these works, a κ-bit string commitment is implemented by sending $O(1)$ group elements and computing $O(1)$ exponentiations in a DDH group of size $2^{O(\kappa)}$. We start this section by presenting a κ-bit UC-secure string commitment protocol in the $\mathcal{F}_{\text{OT}}$-hybrid model where the total communication complexity of each phase (including communication with the OT oracle) is $\kappa(1 + o(1))$. The above implies that if OT exists (in the plain model), then there is a UC-secure protocol for an N-bit string commitment in the CRS model which uses only $N + o(N)$ bits of communication.

Thus, our construction improves over previous protocols which achieve constant rate, but not rate 1. Using, for example, the DDH-based OT protocol of [34], we can get a rate-1 UC-commitment protocol in

Functionality $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$

$\mathcal{F}_{\mathrm{OT_R}}^{\delta}$ with session identifier sid proceeds as follows, running with parties $P_1, \ldots, P_n$, parameters 1^κ and a real number δ, $0 < \delta < 1$, and an adversary $\mathcal{S}$:

- Upon receiving a message $(\mathsf{sender}, sid, ssid, s, r, x)$ from P_s, where $x \in \{0,1\}^\ell$, record the tuple $(sid, ssid, s, r, x)$. (The length of the strings ℓ is fixed and known to all parties.)
- Upon receiving a message $(\mathsf{receiver}, sid, ssid, s, r)$ from P_r, set $y = x$ with probability δ, and $y = \perp$ with probability $1 - \delta$. Send $(sid, ssid, s, r, y)$ to P_r and $(sid, ssid, s, r)$ to P_r, and halt. (If no $(\mathsf{sender}, sid, ssid, s, r, \ldots)$ message was previously sent, then send nothing to P_r.)

Figure 3: Functionality $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$ for Rabin-OT with noise rate δ.

the CRS model which is quite efficient in practice; alternatively, if we wish to obtain a construction in the single global CRS model, we may instead start with the OT protocols given in [10, 1]. We then address the setting where multiple UC commitments need to be realized, showing again a rate-1 construction where, in particular, the number of calls to the OT oracle is independent of the number of UC commitments required. We conclude the section with concrete efficiency analysis of our constructions.

On the "optimality" of our construction. We note that our construction achieves essentially "optimal" rate. In any statistically binding commitment scheme as with our construction, the commit phase communication must be at least the message size. Moreover, any static UC secure commitment scheme must be equivocable, since the simulator for an honest sender does not know the message during the commit phase, and yet must be able to provide openings to any message. Therefore the communication in the decommit phase must be at least the message size, via an argument similar to the lower bound on secret key size in non-committing encryption [32].

3.1 Main construction

Our idea is to use a simple code-based generalization of the standard construction of commitment from δ-Rabin-string-OTs [11, 26, 24, 18]. Our key observation is that the use of a rate-1 encoding scheme with a judicious choice of parameters yields a rate-1 construction of UC commitments. We start off with the following reduction.

Rate-1 Rabin-OT from OT. We first show an efficient realization of Rabin-OT for a given $\delta \in (0,1)$, denoted $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$, in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model, making black-box use of a PRG.

Lemma 1 (Rabin-OT from OT [5, 11, 26, 24])**.** *Let $G : \{0,1\}^{\kappa_{\mathrm{prg}}} \to \{0,1\}^\ell$ be a secure PRG, and let $\delta \in (0,1)$ such that $1/\delta$ is an integer. Then, there exists a protocol which UC-realizes a single instance of $\mathcal{F}_{\mathrm{OT_R}}^{\delta}[\ell]$ in the $\mathcal{F}_{\mathrm{OT}}[\kappa_{\mathrm{prg}}]$-hybrid model such that:*

- *The protocol has total communication complexity at most $\ell + (1/\delta) + 3\kappa_{\mathrm{prg}} \cdot 1/\delta$ bits, including communication with $\mathcal{F}_{\mathrm{OT}}[\kappa_{\mathrm{prg}}]$.*
- *The protocol makes at most $1/\delta$ calls to the $\mathcal{F}_{\mathrm{OT}}[\kappa_{\mathrm{prg}}]$ functionality and requires each party to make a single invocation of G.*

The protocol works by implementing $\mathcal{F}_{\mathrm{OT_R}}^{\delta}[\ell]$ in the $\mathcal{F}_{\mathrm{OT}}^{N}[\ell]$-hybrid model for $N = 1/\delta$. Then $\mathcal{F}_{\mathrm{OT}}^{N}[\ell]$ is realized in the $\mathcal{F}_{\mathrm{OT}}[\kappa_{\mathrm{prg}}]$-hybrid model. The proof is presented in Appendix B.

Encoding scheme Enc

Parameters: n', d, n such that $n' > d > n$.
Input: $m \in \{0,1\}^\ell$ for any $\ell > n \log(n + n')$.

- Parse $m \in \{0,1\}^\ell$ as $(m_1, \ldots, m_n) \in \mathbb{F}^n$ where $\mathbb{F}$ is such that $\log |\mathbb{F}| = \ell/n$.
- Let $e_1, \ldots, e_n$ and $\alpha_1, \ldots, \alpha_{n'}$ be $(n + n')$ distinct elements in $\mathbb{F}$.
- Pick random polynomial p of degree d such that $m_i = p(e_i)$ for all $i \in [n]$.
- Output encoding $m' = (p(\alpha_1), \ldots, p(\alpha_{n'})) \in \mathbb{F}^{n'}$.

Figure 4: A rate-1 encoding scheme based on the multi-secret sharing scheme of [17].

Rate-1 UC-commitments from Rabin-OT. The construction is presented in the following lemma. Further construction and proof details can be found in Appendix C.

Lemma 2. *Let σ be a statistical security parameter, and let n be such that there exists $\epsilon \in (0, 1/2)$ satisfying $n^{1-2\epsilon} = \sigma^{\Omega(1)}$. Then, for $\delta = (2n^\epsilon + 4)^{-1}$, and any $\ell > n \log(2n + 2n^{1-\epsilon})$, there exists a protocol that statistically UC realizes (cf. Definition 2) a single instance of $\mathcal{F}_{\mathrm{MCOM}}[\ell]$ in the $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\ell/n]$-hybrid model in the presence of static adversaries such that:*

- *The protocol has communication complexity $\ell(1 + 2n^{-\epsilon})$ bits in each phase, including communication with $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\ell/n]$.*
- *The protocol makes $n(1 + 2n^{-\epsilon})$ calls to the $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\ell/n]$ functionality.*

Proof. The protocol uses the randomized encoding scheme Enc described in Figure 4 with parameters n as in the Lemma, and $n' = n + 2n^{1-\epsilon}$ and $d = n + n^{1-\epsilon} - 1$. Note that $\delta = (d+1-n)/2n'$. Scheme Enc takes as input $m \in \{0,1\}^\ell$ and parses them as n elements from a field $\mathbb{F}$ and satisfies the following properties (see Lemma 10):

- it has rate $1 + 2n^{-\epsilon}$;
- any $(d + 1 - n)/n' = 2\delta$ fraction of the symbols reveal no information about the encoded message[2];
- any encodings of two distinct messages differ in $\Delta \stackrel{\text{def}}{=} n' - d$ positions (and we can efficiently correct $\Delta/2$ errors);

The construction realizing $\mathcal{F}_{\mathrm{MCOM}}[\ell]$ in the $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\ell/n]$-hybrid model is described in Figure 5. We first analyze the protocol's complexity:

Communication. In the commit phase, the sender transmits the encoding, i.e., $n(1 + 2n^{-\epsilon})$ symbols of $\mathbb{F}$ via $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\ell/n]$. Since $\log |\mathbb{F}| = \ell/n$, the communication complexity is $(n + 2n^{1-\epsilon}) \cdot \ell/n = \ell(1 + 2n^{-\epsilon})$ bits. In the reveal phase, the sender sends the encoding in the clear. It follows from the calculations above that the communication complexity of this phase is also $\ell(1 + 2n^{-\epsilon})$ bits.

Computation. In the commit phase, the sender makes $n(1 + 2n^{-\epsilon})$ calls to $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\ell/n]$.

We now turn to the proof of security. Note that $\delta = O(n^{-\epsilon})$ while $\Delta = O(n^{1-\epsilon})$. Simulating when no party is corrupted or both parties are corrupted is straightforward. We briefly sketch how we simulate a corrupted sender and a corrupted receiver:

Corrupt sender. Here the simulator extracts the committed value by looking at the corrupted codeword $\mathbf{c}$ that P_s sends to the ideal OT functionality and compute the unique codeword $\mathbf{c}^*$ that differs from $\mathbf{c}$ in at most $\Delta/2$ positions. In addition, the simulator reveals each symbol of $\mathbf{c}$ to the honest receiver with probability δ.

[2]We actually require a slightly stronger property to achieve equivocation, namely, that we can efficiently extend a random partial assignment to less than 2δ fraction of the symbols to an encoding of any message.

Realizing $\mathcal{F}_{\mathrm{MCOM}}$ in the $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$-hybrid model

Let $\mathsf{Enc} : \mathbb{F}^n \to \mathbb{F}^{n'}$ be a randomized encoding scheme as in Figure 4. (See also Appendix C.1.)

Commit Phase.

1. Upon receiving input $(\mathsf{commit}, sid, ssid, s, r, m)$ with ℓ-bit input m, party P_s parses m as $(m_1, \ldots, m_n) \in \mathbb{F}^n$. It then computes $m' = (m'_1, \ldots, m'_{n'}) \leftarrow \mathsf{Enc}(m)$.
2. For each $j \in [n']$:
   - P_s sends $(\mathsf{sender}, sid, ssid \circ j, s, r, m'_j)$ to $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$.
   - P_r sends $(\mathsf{receiver}, sid, ssid \circ j, s, r)$ to $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$.
   - P_s and P_r receive $(sid, ssid \circ j, s, r)$ and $(sid, ssid \circ j, s, r, y_j)$ respectively from $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$.
3. P_s keeps state $(sid, ssid, s, r, m, m')$.
4. P_r keeps state $(sid, ssid, s, r, \{y_j\}_{j \in [n']})$, and outputs $(\mathsf{receipt}, sid, ssid, s, r)$. Also, P_r ignores any later commitment messages with the same $(sid, ssid)$ from P_s.

Opening Phase.

1. Upon input $(\mathsf{reveal}, sid, ssid, P_s, P_r)$, party P_s sends $(sid, ssid, m')$, where $m' \in \mathbb{F}^{n'}$, to P_r. Let P_r receive $(sid, ssid, \widetilde{m}')$, where $\widetilde{m}' = (\widetilde{m}'_1, \ldots, \widetilde{m}'_{n'})$.
2. Let J denote the set $\{j \ : \ y_j \neq \perp\}$. P_r outputs $\perp$ if any of the following checks fail:
   - $\widetilde{m}'$ is an (error-free) codeword;
   - for all $j \in J$, it holds that $y_j = \widetilde{m}'_j$.

   If both conditions hold, P_r decodes $\widetilde{m}'$ to obtain $\widetilde{m}$, and outputs $(\mathsf{reveal}, sid, ssid, s, r, \widetilde{m})$.

Figure 5: A statistically UC-secure protocol for $\mathcal{F}_{\mathrm{MCOM}}$ in the $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$-hybrid model.

If $\mathbf{c}$ and $\mathbf{c}^*$ agree on all the positions that are revealed, then the committed value is the message corresponding to $\mathbf{c}^*$; else the committed value is $\perp$.

Next, suppose P_s sends a codeword $\mathbf{c}'$ in the reveal phase. We consider two cases:

- if $\mathbf{c}'$ and $\mathbf{c}$ differ in at most $\Delta/2$ positions, then $\mathbf{c} = \mathbf{c}^*$ and the simulator extracted the correct value;
- otherwise, the honest receiver accepts with probability at most $(1-\delta)^{\Delta/2}$, which is negligible in σ.

Corrupt receiver. In the commit phase, the simulator acts as the ideal OT functionality and for each symbol of the encoding, decides with probability δ whether to send (and, thereby fix) a random element of $\mathbb{F}$ as that symbol to the receiver.

Next, the simulator receives the actual message m in the reveal phase. We consider two cases:

- As long as less than a 2δ fraction of the symbols are transmitted in the simulated commit phase above, the simulator can efficiently extend a random partial assignment implied by the transmitted symbols to the encoding of m;
- otherwise, the simulation of the reveal phase fails with probability at most $e^{-n'\delta/3}$, which is negligible in σ.

□

Putting things together:

Theorem 3 (Rate-1 UC commitments from OT)**.** *Let κ be a computational security parameter, and let $\alpha \in (0, 1/2)$. Then, there is a protocol which UC-realizes a single instance of $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ using κ^{α} calls to $\mathcal{F}_{\mathrm{OT}}[\kappa^{\alpha}]$ and a black-box use of a PRG, where the total communication complexity of each phase (including communication with $\mathcal{F}_{\mathrm{OT}}$) is $\kappa(1 + o(1))$.*

Proof. We set $\ell = \kappa$ and $\sigma = \kappa$. Then we pick $n, \epsilon \in (0, 1/2)$ such that $n^{1+\epsilon} = \kappa^\alpha/10$. Note that $\sigma, n, \epsilon, \ell$ satisfy conditions of Lemma 2. Further, setting $\kappa_{\text{prg}} = \kappa^\alpha$, also ensures that $O(\kappa_{\text{prg}} n^{1+\epsilon}) = o(\kappa)$. The security proof readily follows from composing the protocols given in the Lemmas 1 and 2. We just need to analyze the complexity of the resulting protocol.

Communication. By Lemma 1, to implement $n + 2n^{1-\epsilon}$ calls to $\mathcal{F}_{\text{OT}_\text{R}}^{\delta}[\kappa/n]$, we need to communicate $(n + 2n^{1-\epsilon})((\kappa/n) + O(\kappa_{\text{prg}} n^{\epsilon})) = \kappa + 2\kappa n^{1-\epsilon} + O(\kappa_{\text{prg}} n^{1+\epsilon})$ bits in the $\mathcal{F}_{\text{OT}}[\kappa_{\text{prg}}]$-hybrid model. For $n, \epsilon, \kappa_{\text{prg}}$ as set above, it follows that the communication cost of this phase is $\kappa(1 + o(1))$ bits in each phase.
Computation. By Lemma 1, to implement the required $n + 2n^{1-\epsilon}$ calls to $\mathcal{F}_{\text{OT}_\text{R}}^{\delta}[\kappa/n]$, we need to make blackbox use of PRG, and additionally $(n + 2n^{1-\epsilon}) \cdot (1/\delta) = 2n^{1+\epsilon} + 8n$, i.e., at most κ^α calls to the $\mathcal{F}_{\text{OT}}[\kappa^\alpha]$ functionality. □

3.2 Multiple commitment instances

Next, we show how to further reduce the computational complexity of the previous construction by using OT extension [2, 23, 24]. Our improvement here extends to the setting where we need to perform a large number of commitments in a single parallel commit phase (with potentially many reveal phases), as with applications involving cut-and-choose. In particular, we show that the number of calls to $\mathcal{F}_{\text{OT}}[\kappa_{\text{prg}}]$ can be made independent of the number of instances of UC commitments required. (Note that such a result does not follow from multiple applications of the protocol implied by Theorem 3.)

Theorem 4. *Let κ be a computational security parameter, and let $\alpha \in (0, 1/2)$. For all $c > 0$, there exists a protocol which UC-realizes κ^c instances of $\mathcal{F}_{\text{MCOM}}[\kappa]$ with rate $1 + o(1)$ that makes κ^α calls to $\mathcal{F}_{\text{OT}}[\kappa^\alpha]$ and a blackbox use of correlation robust hash functions (alternatively, random oracle, or non-blackbox use of one-way functions).*

Proof. We repeat the protocol of Theorem 3 κ^c times to construct κ^c instances of $\mathcal{F}_{\text{MCOM}}[\kappa]$ using $\kappa^{c+\alpha}$ calls to $\mathcal{F}_{\text{OT}}[\kappa^\alpha]$. By Theorem 3, the communication cost of this construction is $\kappa^c(1 + o(1))$. We note that for each instance of this protocol, the commit phase has $o(\kappa)$ communication in addition to the cost involved in communicating with $\mathcal{F}_{\text{OT}}[\kappa^\alpha]$.

We then implement the required $\kappa^{c+\alpha}$ calls to $\mathcal{F}_{\text{OT}}[\kappa^\alpha]$ using the constant rate UC-secure OT extension protocol of [24] which makes blackbox use of correlation robust hash functions (alternatively, random oracle, or non-blackbox use of one-way functions). This implementation requires κ^α calls to the $\mathcal{F}_{\text{OT}}[\kappa^\alpha]$ functionality, and has communication complexity $O(\kappa^{c+2\alpha}) = o(\kappa^{c+1})$ bits for $\alpha \in (0, 1/2)$. Therefore, the total communication complexity of this protocol in each phase (including communication with the $\mathcal{F}_{\text{OT}}[\kappa^\alpha]$ functionality) is $\kappa^c(1 + o(1))$ for $c > 1$.

□

3.3 Concrete efficiency analysis

In this section, we provide an analysis of the concrete efficiency of our protocol, specifically requiring that the statistical security loss be $< 2^{-\sigma}$ for statistical security parameter σ, and the seedlength for PRG be 128. This reflects the state-of-the-art choices for similar parameters in implementations of secure computation protocols. In addition to the communication complexity, we will also be interested in the number of public key operations. (In practice, public-key operations (e.g., modular exponentiation) are (at least) 3-4 orders of magnitude slower than symmetric-key operations (e.g., AES).)

In the concrete instantiation of our UC commitment protocol in the CRS model, we will use (1) the protocol of Nielsen *et al.* [31] for OT extension in the RO model since it has better concrete security (cost $\approx 6 \cdot 128$ bits for each instance of 128-bit OT excluding the "seed" OTs) than the protocol of [24]), and

(2) the protocol of Peikert *et al.* [34] to realize "seed" OTs in the CRS model (with concrete cost per OT instance equal to 5 modular exponentiations and 6 elements in a DDH group of size 256). Note that for realizing 128 instances of $\mathcal{F}_{\mathrm{OT}}[128]$, the cost is $6 \cdot 128 \cdot 256 = 196608$ bits and the number of modular exponentiations is $5 \cdot 128 = 640$.[3] We stress that this cost is independent of parameters ℓ, σ, and number of commitment instances. In the following we summarize the cost of our construction for some parameters. Our costs are calculated by choosing concrete parameters for the encoding scheme Enc used in Lemma 2, and then apply the transformation of Lemma 1, and finally realizing $\mathcal{F}_{\mathrm{OT}}$ using state-of-the-art protocols as discussed above.

For long strings, say of length $\ell = 2^{30}$, and for $\sigma = 30$, we can get concrete rate as low as 1.5^{-1} in each phase. However, the choice of parameters necessitate working over a field $\mathbb{F}$ with $\log |\mathbb{F}| = 2^{19}$. If we work over relatively smaller fields $\mathbb{F}$ with say $\log |\mathbb{F}| = 512$, then the rate of the encoding can be made 3.04 while keeping the total rate of the commit phase (including cost of realizing OTs + OT extension) 12.7^{-1}. Note, however, that there are standard techniques to reduce the communication cost of realizing OTs in our setting. For instance, by replacing Rabin-OT with d-out-of-n' OT (for d, n' as in Figure 4), we may then use standard OT length extension techniques. This however has the drawback that RS encodings need to be performed over large fields, and further the number of public-key operations increases with the number of commitment instances.

Consider the following alternative approach that ports our construction to work with smaller fields, and yet get concrete rate close to 1. First, the sender parses the message m as a matrix where each element of the matrix is now from the field of desired size. Next, the sender performs a row-wise encoding (using Enc) of this matrix, and sends each column of the encoded matrix via $\mathcal{F}^{\delta}_{\mathrm{OT}_{\mathrm{R}}}$. Later in the reveal phase, the sender simply transmits the encoded matrix. As noted earlier, the above approach lets us work over small fields, and the concrete rate would be as good as the concrete rate for encoding each row.

Next, we discuss the cost of our basic construction when committing to short strings. For short strings, say of length $\ell = 512$ (resp. 256) and $\sigma = 20$, while the rate of our reveal phase can be as low as 4.9^{-1} (resp. 8.12^{-1}), the rate of our commit phase can be very high ($\approx 2000^{-1}$). While we concede that this is not very impressive in terms of communication cost, we wish to stress that our constructions do offer a significant computational advantage over the protocols of [28, 4] since we perform only a fixed number of public key operations independent of the number of commitment instances. In Appendix E, we propose efficient constructions to handle commitments over short strings in settings where a large number of such short commitments are used, e.g., in cut-and-choose techniques.

Efficiency in the preprocessing model. Our protocols can be efficiently adapted to the preprocessing model [3, 31], and further, the online phase of our protocol can be made free of cryptographic operations. First, note that any UC commitment protocol can be preprocessed, for example by committing to a random string in the offline model, and sending the real input masked with this random string in the online commit phase. Therefore, the online rate of the *commit* phase of the protocol in the preprocessing model can always be made 1. Next, the online rate in the *reveal* phase of our protocol is exactly the rate of the underlying encoding. Note that in the online reveal phase, we only need the receiver to check the validity of the encoding.

[3] The protocol of [34] requires CRS of size m for m parties (cf. [10]). However, since CRS is a one-time setup, this does not affect our (amortized) communication cost. Alternatively, we could use the DDH based construction of [10] which uses a constant sized (6 group elements) global CRS for all parties and will only mildly increase (by a multiplicative factor ≈ 6) the cost of realizing the "seed" OTs).

4 UC Commitment Extension

As a corollary of our technique above, we start this section by showing a rate-1 construction for *UC commitment length extension*, that is, a UC commitment protocol for a long message using a single ideal commitment for a short message. The extension protocol additionally requires the use of a semi-honest (stand-alone) OT protocol. We then show that the existence of a semi-honest OT protocol is necessary for UC commitment length extension.

4.1 Rate-1 UC commitment length extension

In this setting, we want a secure realization of a single instance of UC commitment on a ℓ-bit string, for $\ell = \mathrm{poly}(\kappa)$, while allowing the parties to access ideal functionality $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ exactly once. We show that UC commitment length extension can be realized with rate $1 - o(1)$.

Theorem 5 (Rate-1 UC commitment length extension)**.** *Let κ be a computational security parameter, and assume the existence of semi-honest stand-alone oblivious transfer. Then, for all $c > 0$, there exists a protocol which UC-realizes a single instance of $\mathcal{F}_{\mathrm{MCOM}}[\kappa^c]$ with rate $1 - o(1)$ and makes a single call to $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$.*

Proof. The desired protocol is obtained by using the results of [15, 9] to implement the necessary calls to the OT functionality in a protocol obtained by composing protocols of Lemma 2 and Lemma 1.

Using a single call to $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$, we can generate a uniformly random string (URS) of length κ. Interpreting this κ-bit string as a $\kappa^{1/2}$ instances of a $\kappa^{1/2}$-bit URS, and assuming the existence of semi-honest stand-alone OT, one can apply the results of Damgård *et al.* [15], or Choi *et a.* [9] to obtain κ^α instances of $\mathcal{F}_{\mathrm{OT}}[\kappa^\alpha]$ with $p(\kappa^\alpha)$ invocations of a semi-honest stand-alone OT and communication cost $p(\kappa^\alpha)$, where $p(\cdot)$ is some polynomial, as long as $\alpha \leq 1/2$. We set $\alpha \in (0, 1/2)$ such that $p(\kappa^\alpha) = o(\kappa^c)$.

Using Lemma 2 with parameters $\sigma = \kappa$, and n, ϵ such that $n^{1+\epsilon} = \kappa^\alpha/10$, and $\ell = \kappa^c$, we can UC-realize $\mathcal{F}_{\mathrm{MCOM}}[\kappa^c]$ by making $n + 2n^{1-\epsilon}$ calls to $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\kappa^c/n]$ with $\delta = (2n^\epsilon + 4)^{-1}$. Then, setting $\kappa_{\mathrm{prg}} = \kappa^\alpha$, we use Lemma 1 to UC-realize these $n + 2n^{1-\epsilon}$ calls to $\mathcal{F}^{\delta}_{\mathrm{OT_R}}[\kappa^c/n]$ with communication complexity $(n + 2n^{1-\epsilon}) \cdot ((\kappa^c/n) + (1/\delta) + 3\kappa^\alpha \cdot (1/\delta))$ while making $2n^{1+\epsilon} + 8n$ calls to $\mathcal{F}_{\mathrm{OT}}[\kappa_{\mathrm{prg}}]$.

Thus, for parameters $n, \epsilon, \kappa_{\mathrm{prg}}$ as described above, we see that the communication complexity is $\kappa^c(1 + o(1))$ while making (at most) κ^α calls to $\mathcal{F}_{\mathrm{OT}}[\kappa^\alpha]$. As described in the previous paragraph, these κ^α calls to $\mathcal{F}_{\mathrm{OT}}[\kappa^\alpha]$ can be implemented with communication cost $o(\kappa^c)$. Therefore, a single instance of $\mathcal{F}_{\mathrm{MCOM}}[\kappa^c]$ can be realized with communication cost $\kappa^c(1 + o(1))$ in each phase. □

For any setup where it is possible to construct UC-secure commitments on κ-bit strings (i.e., realize $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$), then assuming the existence of semi-honest stand-alone oblivious transfer, Theorem 5 implies that it is possible to realize UC-secure commitments on strings of arbitrary length (in particular, on κ-bit strings) with rate $1 - o(1)$ in that model. We explicitly state this for the CRS model, where it is known that a protocol for UC commitments in the CRS model implies the existence of semi-honest stand-alone oblivious transfer [15].

Corollary 6. *If UC commitments exist in the CRS model, then they exist with rate $1 - o(1)$.*

4.2 UC commitment length extension implies OT

We now show that the existence of semi-honest stand-alone OT is necessary for the result above.

Theorem 7. *Let κ be a computational security parameter, and suppose there exists a protocol in which at most one party is allowed to make (at most) a single call to $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ to UC-realize a single instance of $\mathcal{F}_{\mathrm{MCOM}}[3\kappa]$. Then there exists a protocol for semi-honest stand-alone OT.*

Here we present only a proof sketch. The full proof can be found in Appendix D.

Proof. We begin with a proof (sketch) for a weaker statement, namely, that UC commitment length extension from κ bits to 3κ bits implies key agreement. Recall that key agreement is implied by OT.

Key agreement from length extension. Let Π denote the commitment protocol assumed to exist. We construct a bit agreement protocol between two parties, A and B, from Π as follows:

- A commits to a random 3κ-bit string m by acting as the honest sender in an execution of Π, and in addition, sends the query $q \in \{0,1\}^\kappa$ it makes to the short commitment oracle and a random $r \in \{0,1\}^\kappa$;
- B runs the UC straight-line extractor for Π to obtain m.

Both parties then agree on the Goldreich-Levin hard-core bit [20] $b = \langle m, r\rangle$ of m.

We now want to argue that an eavesdropper does not learn anything about b in two steps:

- First, if we ignore the query q, then the view of the eavesdropper is exactly the commitment-phase transcript for Π, which reveals no information about m, which means m has 3κ bits of *information-theoretic* entropy.
- The query q then reveals at most κ bits of information about m. Therefore, even upon revealing q, the message m still has $\approx 2\kappa$ bits of (min-)entropy. Then, the Goldreich-Levin hard-core bit works as a randomness extractor to derive a random bit from m.

Correctness is straightforward. To establish security against an eavesdropper, we crucially use the fact that a UC commitment scheme is equivocal, which allows us to essentially argue that m has 3κ bits of information-theoretic entropy. (Indeed, revealing κ bits of information about a 3κ-bit pseudorandom string could reveal the entire string, as is the case when we reveal the seed used to generate the output of a pseduorandom generator.)

Remark. For technical reasons, we will require that the equivocal simulator can simulate not only the public transcript of the protocol, but also the query q made to the short commitment oracle. The existence of such a simulator does not follow immediately from UC security, since the query q may not be revealed to the malicious receiver and the environment. To handle this issue, we basically proceed via a case analysis:

- If the honest sender always reveals q to the receiver either in the commit or the reveal phase, then the equivocal simulator must be able to simulate the query q since it is part of the public transcript.
- Otherwise, we show in Lemma 11 that a cheating receiver can break the hiding property of the commitment scheme.

We are now ready to show the OT implication.

OT from length extension. In the OT protocol, A holds (b_0, b_1), B holds σ, and B wants to learn b_σ. The protocol proceeds as follows:

- Alice runs two independent executions Π_0, Π_1 of the key agreement protocol for two random strings $m_0, m_1 \in \{0,1\}^{3\kappa}$ in parallel. In addition, A sends

$$z_0 = b_0 \oplus \langle m_0, r_0\rangle, \quad z_1 = b_1 \oplus \langle m_1, r_1\rangle.$$

- In the execution Π_σ, B behaves as in the key agreement protocol, which allows him to learn $\langle m_\sigma, r_\sigma\rangle$ and thus recover b_σ. In the other execution, B acts as the honest receiver in an execution of commitment scheme Π.

Correctness follows readily from that of key agreement. We argue security as follows:

- First, we claim that a corrupted semi-honest A does not learn σ. This follows from UC security of the commitment scheme against corrupted senders.
- Next, we claim that a corrupted semi-honest B does not learn $b_{1-\sigma}$. This follows essentially from a similar argument to that for the security of the key agreement protocol with two notable differences: (i) in the execution $\Pi_{1-\sigma}$, B acts as the honest receiver in Π (instead of running the extractor as in the key agreement protocol), and (ii) a semi-honest B learns the coin tosses of the receiver in Π, whereas an eavesdropper for the key agreement protocol does not. Handling (i) is fairly straightforward albeit a bit technical; to handle (ii), we simply use the fact that the commitment phase transcript reveals no information about the committed value, even given the coin tosses of the honest receiver.

□

Acknowledgments. We thank the anonymous reviewers of Eurocrypt 2014 for their helpful comments. We would also like to thank Luís Brandão for alerting us to errors in our cost calculations.

References

[1] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Cline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In *Asiacrypt*, 2013.

[2] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 479–488. ACM Press, May 1996.

[3] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology — Eurocrypt 2011*, volume 6632 of *LNCS*, pages 169–188. Springer, 2011.

[4] Olivier Blazy, Celine Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In *ACNS*, 2013.

[5] G. Brassard, C. Crepeau, and J.-M. Robert. Information theoretic reduction among disclosure problems. In *FOCS*, pages 168–173, 1986.

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

[7] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology — Crypto 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, 2001.

[8] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503. ACM Press, May 2002.

[9] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 387–402. Springer, 2009.

[10] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global crs. In *PKC*, pages 73–88, 2013.

[11] Claude Crépeau. Equivalence between two flavours of oblivious transfers. In Carl Pomerance, editor, *Advances in Cryptology — Crypto '87*, volume 293 of *LNCS*, pages 350–354. Springer, 1988.

[12] Ivan Damgård, Bernardo David, Irene Giacomelli, and Jesper Buus Nielsen. Homomorphic uc commitments in uc. Manuscript., 2013.

[13] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 426–437. ACM Press, June 2003.

[14] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology — Crypto 2002*, volume 2442 of *LNCS*, pages 581–596. Springer, 2002.

[15] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. On the necessary and sufficient assumptions for UC computation. In *7th Theory of Cryptography Conference — TCC 2010*, volume 5978 of *LNCS*, pages 109–127. Springer, 2010.

[16] Marc Fischlin, Benoit Libert, and Mark Manulis. Non-interactive and reusable universally composable string commitments with adaptive security. In *Asiacrypt*, pages 468–485, 2011.

[17] Matthew Franklin and Moti Yung. Communication complexity of secure computation. In *STOC*, pages 699–710, 1992.

[18] T. Frederiksen, T. Jakobsen, J. Nielsen, P. Nordholt, and C. Orlandi. Minilego: Efficient secure two-party computation from general assumptions. In *Eurocrypt*, pages 537–556, 2013.

[19] Eiichiro Fujisaki. A framework for efficient fully-equipped UC commitments. In *ePrint 2012/379*, 2012.

[20] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32. ACM Press, May 1989.

[21] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology — Eurocrypt 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[22] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. On the (im-)possibility of extending coin toss. In Serge Vaudenay, editor, *Advances in Cryptology — Eurocrypt 2006*, volume 4004 of *LNCS*, pages 504–521. Springer, 2006.

[23] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology — Crypto 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, 2003.

[24] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, 2008.

[25] Charanjit Jutla and Arnab Roy. Shorter quasi-adaptive nizk proofs for linear subspaces. In *Asiacrypt*, pages 1–20, 2013.

[26] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[27] Daniel Kraschewski. Complete primitives for information-theoretically secure two-party computation. . Retrieved Oct 14, 2013, 2013.

[28] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In *Advances in Cryptology — Eurocrypt 2011*, volume 6632 of *LNCS*, pages 446–466. Springer, 2011.

[29] Yehuda Lindell and Hila Zarosim. On the feasibility of extending oblivious transfer. In *TCC*, pages 519–538, 2013.

[30] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Cryptographic complexity classes and computational intractability assumptions. In *ICS*, pages 266–289, 2010.

[31] Jesper Nielsen, Peter Nordholt, Claudio Orlandi, and Sai Seshank Burra. A new approach to practical active-secure two-party computation. In *Crypto*, pages 681–700, 2012.

[32] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology — Crypto 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, 2002.

[33] Ryo Nishimaki, Eiichiro Fujisaki, and Keisuke Tanaka. An eficient non-interactive universally composable string-commitment scheme. In *IEICE Transactions*, pages 167–175, 2012.

[34] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, 2008.

A Model and Definitions (cont'd)

In this section we provide a brief description of the UC security model. Most of the material in this section is standard and we follow the presentation in [28, 8].

Two distribution ensembles $\{X(\kappa,a)\}_{\kappa\in\mathbb{N},a\in\{0,1\}^*}$ and $\{Y(\kappa,a)\}_{\kappa\in\mathbb{N},a\in\{0,1\}^*}$ are computationally indistinguishable, denoted $\{X(\kappa,a)\} \stackrel{c}{\equiv} \{Y(\kappa,a)\}$, if for every non-uniform polynomial-time distinguisher D there exists a negligible function μ such that for all $a \in \{0,1\}^*$ and $\kappa \in \mathbb{N}$, $|\Pr[D(X(\kappa,a)) = 1] - \Pr[D(Y(\kappa,a)) = 1]| \leq \mu(\kappa)$. Similary, two distribution ensembles $\{X(\sigma,a)\}_{\sigma\in\mathbb{N},a\in\{0,1\}^*}$ and $\{Y(\sigma,a)\}_{\sigma\in\mathbb{N},a\in\{0,1\}^*}$ are statistically indistinguishable, denoted $\{X(\sigma,a)\} \stackrel{s}{\equiv} \{Y(\sigma,a)\}$, if for every non-uniform (possibly unbounded) distinguisher D there exists a negligible function μ such that for all $a \in \{0,1\}^*$ and $\sigma \in \mathbb{N}$, $|\Pr[D(X(\sigma,a)) = 1] - \Pr[D(Y(\sigma,a)) = 1]| \leq \mu(\sigma)$. We write $\{X(\sigma,a)\} \equiv \{Y(\sigma,a)\}$ if the distributions are identical, in which case we say that the distributions are perfectly indistinguishable.

Universal Composability [6]. Universal composability is a definition of security that considers a stand-alone execution of a protocol in a special setting involving an environment machine $\mathcal{Z}$, in addition to the

honest parties and adversary. As with the classical definition of secure computation, ideal and real models are considered where a trusted party carries out the computation in the ideal model and the real protocol is run in the real model. The environment adaptively chooses the inputs for the honest parties, interacts with the adversary throughout the computation, and receives the honest parties' outputs. Security is formulated by requiring the existence of an ideal-model simulator $\mathcal{S}$ so that no environment $\mathcal{Z}$ can distinguish between the case that it runs with the real adversary $\mathcal{A}$ in the real model and the case that it runs with the ideal-model simulator $\mathcal{S}$ in the ideal model. We use the formulation of the definition of UC security that appears in [8].

In slightly more detail, we denote by $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, z)$ (resp. $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\sigma, z)$) the output of the environment $\mathcal{Z}$ modeled by a probabilistic polynomial-time ITM (resp. unbounded ITM) with input z after an ideal execution with the ideal adversary (simulator) $\mathcal{S}$ modeled by a probabilistic polynomial-time ITM (resp. unbounded ITM) and functionality $\mathcal{F}$, with security parameter κ (resp. statistical security parameter σ). Furthermore, we denote by $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ (resp. $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\sigma, z)$) the output of environment $\mathcal{Z}$ modeled by a probabilistic polynomial-time ITM (resp. unbounded ITM) with input z after a real execution of the protocol π with adversary $\mathcal{A}$ modeled by a probabilistic polynomial-time ITM (resp. unbounded ITM), with security parameter κ (resp. statistical security parameter σ).

Our protocols will be run in a hybrid model where the parties have access to an ideal functionality $\mathcal{G}$. In such a case, according to the definition in [8], all messages between the parties and the ideal functionality are delivered by the adversary. We consider a model with ideally authenticated channels, and so the adversary is allowed to read the messages sent but cannot modify them. In contrast to messages sent between the parties which can be read by the adversary, messages sent between the parties and the ideal functionality are comprimised of a *public header* and *private content*. The public header contains information that is not secret (like the message type, session identifier, the sending and receiving party), whereas the private content contains information that the adversary is not allowed to learn like the parties' private inputs. See [8] for more details. We denote an execution of π in such a model with probablistic polynomial-time (resp. unbounded) ITMs $\mathcal{A}$,$\mathcal{S}$,$\mathcal{Z}$ by $\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ (resp. $\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(\sigma, z)$).

We now define when a protocol π computationally UC realizes a functionality $\mathcal{F}$ in the $\mathcal{G}$ hybrid model.

Definition 1. Let $n \in \mathbb{N}$. Let $\mathcal{F}$ be an ideal functionality and let π be an n-party protocol. We say that π computationally UC realizes $\mathcal{F}$ if there exists an probabilistic polynomial-time ideal-process adversary $\mathcal{S}$ such that for any probabilistic polynomial-time adversary $\mathcal{A}$ and for any non-uniform polynomial-time environment $\mathcal{Z}$,

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*} \stackrel{c}{\equiv} \{\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$$

Next, we define when a protocol π statistically UC realizes a functionality $\mathcal{F}$ in the $\mathcal{G}$ hybrid model.

Definition 2. Let $n \in \mathbb{N}$. Let $\mathcal{F}$ be an ideal functionality and let π be an n-party protocol. We say that π UC realizes $\mathcal{F}$ if there exists an probabilistic unbounded ideal-process adversary $\mathcal{S}$ such that for any probabilistic unbounded adversary $\mathcal{A}$ and for any non-uniform unbounded environment $\mathcal{Z}$,

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\sigma, z)\}_{\sigma\in\mathbb{N},z\in\{0,1\}^*} \stackrel{s}{\equiv} \{\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(\sigma, z)\}_{\sigma\in\mathbb{N},z\in\{0,1\}^*}$$

Finally, we define when a protocol π perfectly UC realizes a functionality $\mathcal{F}$ in the $\mathcal{G}$ hybrid model.

Definition 3. Let $n \in \mathbb{N}$. Let $\mathcal{F}$ be an ideal functionality and let π be an n-party protocol. We say that π UC realizes $\mathcal{F}$ if there exists an probabilistic unbounded ideal-process adversary $\mathcal{S}$ such that for any probabilistic unbounded adversary $\mathcal{A}$ and for any non-uniform unbounded environment $\mathcal{Z}$,

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\sigma,z)\}_{\sigma\in\mathbb{N},z\in\{0,1\}^*} \equiv \{\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(\sigma,z)\}_{\sigma\in\mathbb{N},z\in\{0,1\}^*}$$

Note that when $\mathcal{G}$ is the empty functionality, the $\mathcal{G}$-hybrid model is actually the real model.

The importance of the above definitions is a composition theorem that states that any protocol that is universally composable is secure when run concurrently with many other arbitrary protocols; see [6, 8] for discussions and definitions.

B Realizations of Variants of UC-Secure Oblivious Transfer

B.1 Proof of Lemma 1

Proof. First, we design a UC-secure protocol for $\mathcal{F}^{\delta}_{\text{OT}_\text{R}}[\ell]$ in the $\mathcal{F}^{N}_{\text{OT}}[\kappa_{\text{prg}}]$-hybrid model, where $N = 1/\delta$, making blackbox use of the pseudorandom generator G. The protocol uses ideas from [24], and is presented in Figure 6. (A proof of security of this protocol appears in Appendix B.2.) From the protocol description in Figure 6, it follows that for $\delta = 1/N$,

$$\mathsf{cc}(\mathcal{F}^{\delta}_{\text{OT}_\text{R}}[\ell]) = N + \ell + \mathsf{cc}(\mathcal{F}^{N}_{\text{OT}}[\kappa_{\text{prg}}]). \tag{1}$$

Next, we design a UC-secure protocol for $\mathcal{F}^{N}_{\text{OT}}[\kappa_{\text{prg}}]$ in the $\mathcal{F}_{\text{OT}}[\kappa_{\text{prg}}]$-hybrid model. For this we use the information-theoretically secure transformation of Brassard et al. [5] described in Figure 7. (A proof of security appears in Appendix B.3.) From the protocol description in Figure 7, it follows that

$$\mathsf{cc}(\mathcal{F}^{N}_{\text{OT}}[\kappa_{\text{prg}}]) = (N-1)\cdot \mathsf{cc}(\mathcal{F}_{\text{OT}}[\kappa_{\text{prg}}]). \tag{2}$$

Then we obtain the desired protocol satisfying Lemma 1 by composing the UC-secure protocols above, and the proof of security is obtained by a straightforward application of the composition theorem. From Equations 1 and 2, it follows that

$$\mathsf{cc}(\mathcal{F}^{\delta}_{\text{OT}_\text{R}}[\ell]) = \ell + N + (N-1)\cdot \mathsf{cc}(\mathcal{F}_{\text{OT}}[\kappa_{\text{prg}}]). \tag{3}$$

It is easy to see that the total communication complexity of the protocol, including communication with $\mathcal{F}_{\text{OT}}[\kappa_{\text{prg}}]$ (which induces a factor 3) is $\ell + (1/\delta) + 3\kappa_{\text{prg}}\cdot(1/\delta))$, and that the protocol makes at most $1/\delta$ calls to $\mathcal{F}_{\text{OT}}[\kappa_{\text{prg}}]$ and each party makes a single invocation of G. □

B.2 Realizing $\mathcal{F}^{\delta}_{\text{OT}_\text{R}}[\ell]$ in the $\mathcal{F}^{N}_{\text{OT}}[\kappa_{\text{prg}}]$-Hybrid Model

The protocol realizing $\mathcal{F}^{\delta}_{\text{OT}_\text{R}}[\ell]$ in the $\mathcal{F}^{N}_{\text{OT}}[\kappa_{\text{prg}}]$-hybrid model is described in Figure 6.

Lemma 8. *Let parameters δ and N satisfy the following relation: there exists an integer a such that $a \in [N]$, and $\delta = a/N$. Further let $G : \{0,1\}^{\kappa_{\text{prg}}} \to \{0,1\}^{\ell}$ be a pseudorandom generator. Then the protocol in Figure 6 computationally UC realizes (cf. Definition 1) the $\mathcal{F}^{\delta}_{\text{OT}_\text{R}}[\ell]$ functionality in the $\mathcal{F}^{\delta}_{\text{OT}_\text{R}}[\kappa_{\text{prg}}]$-hybrid model in the presence of static adversaries.*

Proof. We give the description of the simulator.

The simulator $\mathcal{S}$:

Realizing $\mathcal{F}_{\mathrm{OT_R}}^{\delta}[\ell]$ in the $\mathcal{F}_{\mathrm{OT}}^{N}[\kappa_{\mathrm{prg}}]$-hybrid model

The parameters δ and N satisfy the following relation: there exists an integer $a \in [N]$ such that $\delta = a/N$ holds. Let $G : \{0,1\}^{\kappa_{\mathrm{prg}}} \to \{0,1\}^{\ell}$ be a pseudorandom generator.

- Given input $(\mathsf{sender}, sid, ssid, s, r, x)$, party P_s chooses $K, K_1, \ldots, K_N \in \{0,1\}^{\kappa_{\mathrm{prg}}}$ at random subject to $|\{K_j : (K_j = K) \wedge j \in [N]\}| = a$.
- P_s sends $(\mathsf{sender}, sid, ssid, s, r, K_1, \ldots, K_N)$ to $\mathcal{F}_{\mathrm{OT}}^{N}$.
- P_r chooses random $i \in [m]$ and sends $(\mathsf{receiver}, sid, ssid, s, r, i)$ to $\mathcal{F}_{\mathrm{OT}}^{N}$, and receives back K_i.
- Upon receiving $(sid, ssid, s, r)$ from $\mathcal{F}_{\mathrm{OT}}^{N}$, P_i sends the N-bit characteristic vector of the set $A = \{j : K_j = K\}$ along with the value $c = G(K) \oplus x$ to P_r.
- Upon receiving characteristic vector of A from P_s, party P_r generates its output z as $z = c \oplus G(K_i)$ if $i \in A$, else $z = \perp$.

Figure 6: A computationally UC-secure protocol for $\mathcal{F}_{\mathrm{OT_R}}^{\delta}[\ell]$ in the $\mathcal{F}_{\mathrm{OT}}^{N}[\kappa_{\mathrm{prg}}]$-hybrid model.

- **Simulating the communication with $\mathcal{Z}$:** Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.
- **Simulating the protocol when P_s is corrupted and P_r is honest:** Acting as the ideal functionality $\mathcal{F}_{\mathrm{OT}}^{N}[\kappa_{\mathrm{prg}}]$, the simulator $\mathcal{S}$ receives $(\mathsf{sender}, sid, ssid, s, r, K_1, \ldots, K_N)$ from $\mathcal{A}$ controlling P_s, and sends back $(sid, ssid, s, r)$ to $\mathcal{A}$. Then, $\mathcal{S}$ receives the N-bit characterstic vector of a set A and a value $c \in \{0,1\}^{\ell}$ from $\mathcal{A}$. If $|A| \neq a$, then $\mathcal{S}$ sends $(\mathsf{sender}, sid, ssid, s, r, \perp)$ to $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$, and terminates the simulation. Else, it picks a random index $j \in A$ and sends $(\mathsf{sender}, sid, ssid, s, r, c \oplus G(K_j))$ to $\mathcal{F}_{\mathrm{OT_R}}^{\delta}[\ell]$, and terminates the simulation.
- **Simulating the protocol when P_r is corrupted and P_s is honest:** Acting as the ideal functionality $\mathcal{F}_{\mathrm{OT}}^{N}$, the simulator $\mathcal{S}$ receives $(\mathsf{receiver}, sid, ssid, s, r, i)$ from $\mathcal{A}$ controlling P_r. $\mathcal{S}$ chooses a random $K \in \{0,1\}^{\kappa_{\mathrm{prg}}}$ and sends $(sid, ssid, K)$ to P_r. Upon receiving $(sid, ssid, s, r, y)$ from $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$, the simulator checks whether $y = \perp$ holds. If $y = \perp$, then $\mathcal{S}$ chooses random $c \in \{0,1\}^{\ell}$, and a random set $A \subseteq [N]$ such that $i \notin A$ and $|A| = a$. Else, $\mathcal{S}$ sets $c = G(K) \oplus y$, and chooses a random set $A \subseteq [N]$ such that $i \in A$ and $|A| = a$. $\mathcal{S}$ sends the N-bit characteristic vector of the set A along with the value c to $\mathcal{A}$.

  Simulation in the case when both P_s and P_r are honest is straightforward.

Analysis of the simulation: Denoting the protocol in Figure 6 by π, and recalling that it runs in the $\mathcal{F}_{\mathrm{OT}}^{N}$-hybrid model, we need to prove that the simulator $\mathcal{S}$ described above is such that for for every $\mathcal{A}$, and for every $\mathcal{Z}$,

$$\{\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{OT_R}}^{\delta}, \mathcal{S}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*} \stackrel{c}{\equiv} \{\mathrm{HYBRID}_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\mathrm{OT}}^{N}}\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}.$$

The case when $\mathcal{A}$ corrupts P_s is easy to handle. In this case, note that $\mathcal{A}$ does not receive any message in the protocol other than $(sid, ssid, s, r)$. It is also straightforward to see that the output of honest P_r is identically distributed in both the ideal and hybrid processes.

When $\mathcal{A}$ corrupts P_r, we argue that the simulation is indistinguishable from the hybrid protocol as long as G is a computationally secure pseudorandom generator. More precisely, given an environment $\mathcal{Z}$ which

Realizing $\mathcal{F}_{\mathrm{OT}}^{N}[\ell]$ in the $\mathcal{F}_{\mathrm{OT}}[\ell]$-hybrid model [5]

- Given input $(\mathsf{sender}, sid, ssid, s, r, x_1, \ldots, x_N)$ with $x_1, \ldots, x_N \in \{0,1\}^\ell$, party P_s initializes $y_{1,0} = x_1$, and then for $j = 1$ to $N-2$, performs the following:
  - Choose $r_j \leftarrow \{0,1\}^\ell$ at random.
  - Set $y_{j,1} = r_j$.
  - Set $y_{j+1,0} = x_{j+1} \oplus \big(\bigoplus_{k=1}^{j} r_k\big)$.

  Finally, P_s sets $y_{N-1,1} = x_N \oplus \big(\bigoplus_{k=1}^{N-2} r_k\big)$.
- Given input $(\mathsf{sender}, sid, ssid, s, r, q)$, party P_r computes values $q_1, \ldots, q_{N-1} \in \{0,1\}$ such that for each $j \in [N-1]$, the value q_j equals 0 iff $q_j = q$.
- For each $j \in [N-1]$:
  - P_s sends $(\mathsf{sender}, sid, ssid \circ j, s, r, y_{j,0}, y_{j,1})$ to $\mathcal{F}_{\mathrm{OT}}$.
  - P_r sends $(\mathsf{receiver}, sid, ssid \circ j, s, r, q_j)$ to $\mathcal{F}_{\mathrm{OT}}$.
  - P_s and P_r receive $(sid, ssid \circ j, s, r)$ and $(sid, ssid \circ j, s, r, y_{j,q_j})$ respectively from $\mathcal{F}_{\mathrm{OT}}$.
- P_r sets $y_{N,0} = 0^\ell$ and outputs $z = y_{q,0} \oplus \big(\bigoplus_{k=1}^{q-1} y_{k,1}\big)$.

Figure 7: A perfectly UC-secure protocol for $\mathcal{F}_{\mathrm{OT}}^{N}$ in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model.

distinguishes between the ideal and hybrid processes in which $\mathcal{A}$ corrupts P_r, we demonstrate an adversary $\mathcal{A}_{\text{prg}}$ that can break the pseudorandomness of G. Upon input w which is either random or pseudorandom (i.e., $G(u)$ for some $u \in \{0,1\}^{\kappa_{\text{prg}}}$), $\mathcal{A}_{\text{prg}}$, interacting with $\mathcal{Z}$ and $\mathcal{A}$ corrupting P_r, simulates an execution of $\text{IDEAL}_{\mathcal{F}_{\mathrm{OT_R}}^{\delta}, \mathcal{S}, \mathcal{Z}}$, playing the roles of honest P_s and emulating $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$ honestly, except that when $\perp$ is received from (the simulated) $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$, $\mathcal{A}_{\text{prg}}$ sets $c = w \oplus y$, where y is the honest P_s's input chosen by $\mathcal{Z}$. Finally, $\mathcal{A}_{\text{prg}}$ outputs what $\mathcal{Z}$ outputs.

It is easy to see that when w is random, $\mathcal{A}_{\text{prg}}$'s simulation is identical to the ideal process. On the other hand when w is pseudorandom, $\mathcal{A}_{\text{prg}}$'s simulation is identical to the protocol in the $\mathcal{F}_{\mathrm{OT}}^{N}$-hybrid model described in Figure 6.

Given this, we conclude that the ideal process and the hybrid process are computationally indistinguishable by the fact that G is a computationally secure pseudorandom generator. □

B.3 Realizing $\mathcal{F}_{\mathrm{OT}}^{N}[\ell]$ in the $\mathcal{F}_{\mathrm{OT}}[\ell]$-Hybrid Model

The protocol realizing $\mathcal{F}_{\mathrm{OT}}^{N}[\ell]$ in the $\mathcal{F}_{\mathrm{OT}}[\ell]$-hybrid model is described in Figure 7.

Lemma 9. *The protocol in Figure 7 perfectly UC realizes (cf. Definition 3) the $\mathcal{F}_{\mathrm{OT}}^{N}[\ell]$ functionality in the $\mathcal{F}_{\mathrm{OT}}[\ell]$-hybrid model in the presence of static adversaries.*

Proof. We give the description of the simulator.

The simulator $\mathcal{S}$:

- **Simulating the communication with $\mathcal{Z}$:** Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.

- **Simulating the protocol when P_s is corrupted and P_r is honest:** Acting as the ideal functionality $\mathcal{F}_{\mathrm{OT}}[\ell]$, the simulator $\mathcal{S}$ performs the following actions for all $j \in [N-1]$: $\mathcal{S}$ receives $(\mathsf{sender}, sid, ssid \circ j, s, r, y_{j,0}, y_{j,1})$ from $\mathcal{A}$ controlling P_s, and sends back $(sid, ssid \circ j, s, r)$ to $\mathcal{A}$. Then, $\mathcal{S}$ locally sets $y_{N,0} = 0^\ell$. For all $q \in [N]$, $\mathcal{S}$ computes $x_q = y_{q,0} \oplus \big(\bigoplus_{k=1}^{q-1} y_{k,1}\big)$. Finally, $\mathcal{S}$ sends $(\mathsf{sender}, sid, ssid, s, r, x_1, \ldots, x_N)$ to $\mathcal{F}_{\mathrm{OT}}^N[\ell]$.

- **Simulating the protocol when P_r is corrupted and P_s is honest:** Acting as the ideal functionality $\mathcal{F}_{\mathrm{OT}}[\ell]$, the simulator $\mathcal{S}$ initializes $\mathsf{flag} = 0$, and performs the following actions for $j = 1, \ldots, N-1$:

  - $\mathcal{S}$ receives $(\mathsf{receiver}, sid, ssid \circ j, s, r, q_j)$ from $\mathcal{A}$ controlling P_r.
  - If $\mathsf{flag} = 1$ holds or if ($q_j \neq 0$ and $\mathsf{flag} = 0$ and $j \neq N-1$) holds, then $\mathcal{S}$ chooses random $x'_j, r_j \leftarrow \{0,1\}^\ell$, sets $y_{j,0} = x'_j \oplus \big(\bigoplus_{k=1}^{j-1} r_j\big)$ and $y_{j,1} = r_j$, and sends back $(sid, ssid \circ j, s, r, y_{j,q_j})$ to $\mathcal{A}$.
  - If ($q_j = 0$ and $\mathsf{flag} = 0$) holds, then $\mathcal{S}$ sets values $q = j$ and $\mathsf{flag} = 1$, and sends $(\mathsf{receiver}, sid, ssid, s, r, q)$ to $\mathcal{F}_{\mathrm{OT}}^N[\ell]$ and receives back $(sid, ssid, s, r, x')$. $\mathcal{S}$ then chooses random $r_j \leftarrow \{0,1\}^\ell$, sets $y_{j,0} = x' \oplus \big(\bigoplus_{k=1}^{j-1} r_k\big)$ and $y_{j,1} = r_j$, and sends back $(sid, ssid \circ j, s, r, y_{j,q_j})$ to $\mathcal{A}$.
  - If ($\mathsf{flag} = 0$ and $j = N-1$) holds, then $\mathcal{S}$ sets values $q = N$ and $\mathsf{flag} = 1$, and sends $(\mathsf{receiver}, sid, ssid, s, r, q)$ to $\mathcal{F}_{\mathrm{OT}}^N[\ell]$ and receives back $(sid, ssid, s, r, x')$. $\mathcal{S}$ then chooses random $x'_j \leftarrow \{0,1\}^\ell$, sets $y_{j,0} = x'_j \oplus \big(\bigoplus_{k=1}^{j-1} r_k\big)$ and $y_{j,1} = x' \oplus \big(\bigoplus_{k=1}^{N-2} r_k\big)$, and sends back $(sid, ssid \circ j, s, r, y_{j,q_j})$ to $\mathcal{A}$.

  Simulation in the case when both P_s and P_r are honest is straightforward.

Analysis of the simulation: Denoting the protocol in Figure 7 by π, and recalling that it runs in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model, we need to prove that the simulator $\mathcal{S}$ described above is such that for for every $\mathcal{A}$, and for every $\mathcal{Z}$,

$$\{\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{OT}}^N, \mathcal{S}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*} \stackrel{c}{\equiv} \{\mathrm{HYBRID}_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\mathrm{OT}}}\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}.$$

The case when $\mathcal{A}$ corrupts P_s is easy to handle. In this case, note that $\mathcal{A}$ does not receive any message other than $(sid, ssid \circ j, s, r)$ for all $j \in [N-1]$. It is also straightforward to see that the values $\{x_q\}_{q \in [N]}$ extracted from $\{y_{q,0}, y_{q,1}\}_{q \in [N-1]}$ correspond to the input values used by $\mathcal{A}$. Therefore, the output of honest P_r is identically distributed in both the ideal and hybrid processes.

When $\mathcal{A}$ corrupts P_r, we argue that the simulation is perfectly indistinguishable from the hybrid protocol. First, we note that the variable flag is used to indicate when the simulator extracts the selection $q \in [N]$ from $\mathcal{A}$. More concretely, the variable flag is set to 1 (indicating the selection has been extracted from $\mathcal{A}$) when $\mathcal{A}$ makes query $q_j = 0$ to $\mathcal{F}_{\mathrm{OT}}[\ell]$ for some j. In this case, the extracted selection $q = j$. When $\mathcal{A}$ never queries $q_j = 0$ to $\mathcal{F}_{\mathrm{OT}}[\ell]$ for any $j \in [N-1]$, then this corresponds to the extracted selection $q = N$. Note that the way in which the selection value of $\mathcal{A}$ is extracted corresponds exactly to its input value used in the hybrid protocol.

Next, we note that until iteration $j = q$, the view of $\mathcal{A}$ in both the ideal and hybrid executions is $\{r_k\}_{k \in [q-1]}$, i.e., uniform random values in $\{0,1\}^\ell$. When $q < N$, in iteration $j = q$, the simulator uses the output value x' obtained from $\mathcal{F}_{\mathrm{OT}}^N[\ell]$ to construct the values $y_{j,0}, y_{j,1}$ such that $\mathcal{A}$ is able to learn the output x' from $y_{j,0}$ while the value $y_{j,1}$ is uniformly random and distributed identically as in the hybrid protocol, thus leaking no information about values $\{x_k\}_{k \neq q}$. Likewise when $q = N$, in iteration $j = q$, the simulator uses the output value x' obtained from $\mathcal{F}_{\mathrm{OT}}^N[\ell]$ to construct the values $y_{j,0}, y_{j,1}$ such that $\mathcal{A}$ is able to learn the output x' from $y_{j,1}$ while the value $y_{j,0}$, as in the hybrid protocol, is completely hidden from $\mathcal{A}$, thus leaking no information about values $\{x_k\}_{k \neq q}$.

Note that in iteration $j = q$, the variable flag is set to 1. Once this event occurs, the simulator provides only random values as input to $\mathcal{F}_{\mathrm{OT}}[\ell]$ for all iterations $j > q$. We now argue that the distribution of simulated inputs to $\mathcal{F}_{\mathrm{OT}}[\ell]$ for $j > q$ is identical to the distribution of inputs to $\mathcal{F}_{\mathrm{OT}}[\ell]$ in the hybrid protocol. First, note that $\mathcal{A}$ does not have any information on the value r_q. Then, the next time that $\mathcal{A}$ queries $q_j = 0$ for some $j = p_1 > q$, $\mathcal{A}$ does not learn any information since the value $y_{j,0}$ is a value masked with r_q which the adversary does not know. Therefore, the value $y_{j,0}$ appears completely random from the point of view of $\mathcal{A}$. Also, note that at this point, $\mathcal{A}$ does not know the value of r_{p_1}. This allows us to argue that the next time that $\mathcal{A}$ queries $q_j = 0$ for some $j = p_2 > p_1$, $\mathcal{A}$ does not learn any information since the value $y_{j,0}$ is a value masked with r_{p_1} which the adversary does not know. Therefore, the value $y_{j,0}$ appears completely random from the point of view of $\mathcal{A}$. Once again, we observe that at this point, $\mathcal{A}$ does not know the value of r_{p_2}. In other words, each time the adversary makes a query $q_j = 0$, it misses knowing (a new) value r_j which prevents it from learning any information about the value $x_{j'}$ (which is masked with the pad r_j) from $y_{j',0}, y_{j',1}$, where j' is the next index greater than j such that query $q_{j'} = 0$. We complete the argument by noting that whenever $q_j = 1$ for $j < N - 1$, the adversary simply learns a uniformly random pad r_j that is independent of the values $\{x_k\}_{k\in[N]}$. When $q_j = 1$ for $j = N - 1$, as argued before, the adversary learns x_N iff for all $j \in [N - 1]$, the query $q_j = 1$.

Given this, we conclude that the ideal process and the hybrid process are perfectly indistinguishable. □

C Rate-1 UC-Commitments from Rabin-OT (cont'd)

C.1 An Encoding Scheme based on Multi-Secret Sharing Schemes

Let $m \in \{0,1\}^\ell$ represent the message. We parse $m = (m_1, \ldots, m_n)$ as a vector in $\mathbb{F}^n$. Note that $|m_i| = \ell/n$, and hence $\log |\mathbb{F}| = \ell/n$.

Our randomized encoding scheme Enc maps a message from $\mathbb{F}^n$ to a codeword in $\mathbb{F}^{n'}$. The encoding scheme is a variant of RS codes used in [17] to obtain multi-secret sharing schemes. More concretely, let d be a parameter such that $n < d < n'$. Let $\alpha_1, \ldots, \alpha_{n'}$ and $e_1, \ldots, e_n$ be (publicly known) mutually disjoint set of points in $\mathbb{F}$. Let $p(x)$ be a random degree-d polynomial subject to $p(e_i) = m_i$ for all $i \in [n]$. For $j \in [n']$, let $m'_j = p(\alpha_j)$. We call $m' = (m'_1, \ldots, m'_{n'}) \in \mathbb{F}^{n'}$ as the encoding of $m \in \mathbb{F}^n$, and use the notation $m' \leftarrow \mathsf{Enc}(m)$.

Observe that for any $T \subseteq [n']$ satisfying $|T| > d$, the coordinates $\{m'_j\}_{j\in T}$ can be used to decode m' and recover m. Furthermore, for any $T \subset [n']$ satisfying $|T| \leq t \stackrel{\text{def}}{=} d + 1 - n$, the coordinates $\{m'_j\}_{j\in T}$ do not reveal any information about m. Also, observe that for any two messages $m_0, m_1 \in \mathbb{F}^n$, their respective encodings m'_0 and m'_1 differ in at least $d' \stackrel{\text{def}}{=} n' - d$ locations.

Lemma 10. *For any $n > 0$, $d > n$, $n' > d$, and any $\ell > n \log(n + n')$, and any field $\mathbb{F}$ with $|\mathbb{F}| > n + n'$, there exists a randomized encoding scheme mapping messages from $\mathbb{F}^n$ to codewords in $\mathbb{F}^{n'}$ such that*

- *any $2\delta \stackrel{\text{def}}{=} (d + 1 - n)/n'$ fraction of the symbols of the encoding reveal no information about the encoded message, and furthermore, any random partial assignment to less than 2δ fraction of the symbols can be efficiently extended to an encoding of any message;*
- *any encodings of two distinct messages differ in $\Delta \stackrel{\text{def}}{=} n' - d$ positions (and we can efficiently correct $\Delta/2$ errors).*

Proof. We show that the encoding scheme Enc with parameters n, d, n' is the desired encoding scheme. Note that this is possible as long as $|\mathbb{F}| \geq n + n'$.

From the discussion above, we have that $t/n' = 2\delta$ fraction of the symbols of the encoding reveal no information about the encoded message. It is easy to see from the construction above that any random partial assignment to less than t/n' fraction of the symbols can be efficiently extended to an encoding of any message. More concretely, given a random partial assignment to less than t symbols, then for any given message consisting of n symbols, we can always fit a degree-d polynomial through these $\leq n + t - 1 = d$ symbols.

Lastly, any encoding of two distinct messages differ in $\Delta = d' = n' - d = n^{1-\epsilon} - 6n^{1-2\epsilon} + 1$ positions, and we can efficiently correct $\Delta/2$ errors (say, using decoding algorithms for RS codes). □

C.2 Proof of Lemma 2

We borrow some notation from Appendix C.1. In particular, we use auxiliary variables $t = 2n'\delta$, $d = n + t - 1$, and $d' = n' - d = \Delta$ to simplify the exposition.

Proof. We give the description of the simulator.

The simulator $\mathcal{S}$:

- **Simulating the communication with $\mathcal{Z}$:** Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.

- **Simulating the commit stage when the committer P_s is corrupted and the receiver P_r is honest:** For each $j \in [n']$, acting as the ideal functionality $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$, $\mathcal{S}$ receives $(\mathsf{sender}, sid, ssid \circ j, s, r, m'_j)$ from $\mathcal{A}$ controlling P_s, and sends back $(sid, ssid \circ j, s, r)$ to $\mathcal{A}$.

  Next, $\mathcal{S}$ performs an error correction procedure for Enc on $m' = (m'_1, \ldots, m'_{n'})$ to compute the unique codeword m^* that differs from m' in at most $\Delta/2$ positions. Let $m'' \in \mathbb{F}^n$ be message obtained by decoding codeword m^*, and let p'' be the corresponding degree-d polynomial such that $p''(e_j) = m''_j$ for all $j \in [n]$.

  Then, $\mathcal{S}$ constructs a set J'' by picking each index $j \in [n']$ with probability $\delta = t/2n'$. This process defines variables y'_j for $j \in [n']$ as $y'_j = m'_j$ for $j \in J''$, and $y'_j = \perp$ otherwise.

  If $m'_j = m^*_j$ for all $j \in J''$, then $\mathcal{S}$ sends $(\mathsf{commit}, sid, ssid, s, r, m'' = (m''_1, \ldots, m''_n))$ to $\mathcal{F}_{\mathrm{MCOM}}$, where m'' is expected to be parsed as an element of $\{0,1\}^\ell$. Else, $\mathcal{S}$ sends a dummy commitment $(\mathsf{commit}, sid, ssid, s, r, 0)$ to $\mathcal{F}_{\mathrm{MCOM}}$ (and in this case, we will see that $\mathcal{S}$ will not send anything in the reveal phase).

- **Simulating the decommit stage when P_s is corrupted and P_r is honest:** $\mathcal{S}$ receives $(sid, ssid, \widetilde{m}')$, where $\widetilde{m}' \in \mathbb{F}^{n'}$, from $\mathcal{A}$ controlling P_s. If (1) $\widetilde{m}'$ is an (error-free) codeword, and (2) for all $j \in J''$, it holds that $\widetilde{m}'_j = m'_j$, then $\mathcal{S}$ sends $(\mathsf{reveal}, sid, ssid, s, r)$ to $\mathcal{F}_{\mathrm{MCOM}}$. Otherwise, it does nothing.

- **Simulating the commit stage when P_s is honest and P_r is corrupted:** Upon receiving $(\mathsf{receipt}, sid, ssid, s, r)$ from $\mathcal{F}_{\mathrm{MCOM}}$ the simulator $\mathcal{S}$ interacts with $\mathcal{A}$ controlling P_r by acting as ideal functionality $\mathcal{F}_{\mathrm{OT_R}}^{\delta}$ in the following way: for each $j \in [n']$, $\mathcal{S}$ receives $(\mathsf{receiver}, sid, ssid \circ j, s, r)$ from $\mathcal{A}$, and returns $(sid, ssid \circ j, s, r, y_j)$, where y_j is chosen to be a random element of $\mathbb{F}$ with probability $\delta = t/2n'$, and as $\perp$ with probability $1 - \delta = 1 - t/2n'$.

- **Simulating the decommit stage when P_s is honest and P_r is corrupted:** Upon receiving $(\mathsf{reveal}, sid, ssid, s, r, m)$ from $\mathcal{F}_{\mathrm{MCOM}}$, the simulator $\mathcal{S}$ works as follows.

  Let J denote the set $\{j : y_j \neq \perp\}$. $\mathcal{S}$ computes a degree-d polynomial $p'(x)$ such that (1) $p'(\alpha_i) = y_i$ for $i \in J$, and (2) $p'(e_i) = m_i$ for $i \in [n]$. If such a polynomial cannot be constructed, then $\mathcal{S}$ outputs

fail and terminates the simulation. Otherwise, $\mathcal{S}$ computes $\widetilde{m}'_i = p'(\alpha_i)$ for $i \in [n']$, and hands $(sid, ssid, \widetilde{m}' = (\widetilde{m}'_1, \ldots, \widetilde{m}'_{n'}))$ to $\mathcal{A}$, as it expects to receive from P_s.

Simulation in the case when both P_s and P_r are honest is straightforward.

Analysis of the simulation: Denoting the protocol in Figure 5 by π, and recalling that it runs in the $\mathcal{F}_{\text{OT}_\text{R}}^{\delta}$-hybrid model, we need to prove that the simulator $\mathcal{S}$ described above is such that for for every $\mathcal{A}$, and for every $\mathcal{Z}$,

$$\{\text{IDEAL}_{\mathcal{F}_{\text{MCOM}},\mathcal{S},\mathcal{Z}}(\sigma, z)\}_{\sigma\in\mathbb{N},z\in\{0,1\}^*} \stackrel{s}{\equiv} \{\text{HYBRID}^{\mathcal{F}_{\text{OT}_\text{R}}^{\delta}}_{\pi,\mathcal{A},\mathcal{Z}}\}_{\sigma\in\mathbb{N},z\in\{0,1\}^*}.$$

We analyze separately the case when the $\mathcal{A}$ corrupts P_r, and the case when the $\mathcal{A}$ corrupts P_s.

When $\mathcal{A}$ corrupts P_r, the simulator, acting as the ideal functionality $\mathcal{F}_{\text{OT}_\text{R}}^{\delta}$, chooses symbols of the encoding of the message uniformly at random, and delivers them to $\mathcal{A}$ with probability δ. Let J denote the set of indices corresponding to symbols delivered to $\mathcal{A}$. The distribution of J itself is identical between the ideal and hybrid processes, but the distribution of symbols of the encoding corresponding to indices in J are identical (in this case, uniform and independent of the message m) only as long as $|J| \leq d + 1 - n = t$. This is because as long as $|J| \leq d + 1 - n$, the simulator will be able to compute a degree-d polynomial $p'(x)$ that is consistent both with the n message symbols that will be revealed in the decommit phase, and with the $|J|$ random symbols delivered by $\mathcal{S}$ during the commit phase. When $|J| > d + 1 - n$, $\mathcal{S}$ outputs fail since it is unable to compute a degree-d polynomial consistent with $|J| + n$ interpolation points. Since $\mathcal{A}$ corrupting P_r does not send any messages in the protocol, it follows that the ideal and hybrid processes are indistinguishable as long as $\mathcal{S}$ does not output fail. Since expected value of $|J|$ is $t/2$, by a Chernoff bound, the probability that $|J| > t$, i.e., $\mathcal{S}$ outputs fail is bounded by $e^{-t/6} = e^{-n'\delta/3} = \mathsf{negl}(\sigma)$ for σ as in Lemma 2.

Consider the case when $\mathcal{A}$ corrupts P_s. Observe first that since $\mathcal{A}$ does not receive any messages in the protocol (or the simulation), the view of $\mathcal{A}$ in both the ideal and hybrid processes are identical. Next, observe that the distribution of J'' in the ideal process is identical to the distribution of J in the hybrid process.

It is easy to see that if $\mathcal{A}$ does not reveal an error-free codeword, the output of honest receiver in both the ideal and hybrid world are identical. Therefore, we assume that the $\mathcal{A}$ reveals an error-free codeword. Given the above, the distribution of the ideal process and hybrid process differs only when the value m'' extracted by $\mathcal{S}$ differs from the value $\widetilde{m}$ output by P_r in the hybrid process. The key observation is that the values m'' and $\widetilde{m}$ are valid codewords and hence have minimum distance Δ. This implies that the corrupt sender's encoding m' that it provided in the commit phase differs from $\widetilde{m}'$ that it provides in the reveal phase in at least $\Delta/2$ symbols. Clearly this adversarial strategy succeeds only when none of the differing position occur in J'' (in which case the honest receiver accepts a value different from what the simulator extracted). Recall that $\mathcal{S}$ chooses each position $j \in J''$ with probability δ, as in the real process. Therefore, the probability that none of the $\Delta/2$ differing positions occurred in J'' is $(1-\delta)^{\Delta/2} \leq e^{-\delta\Delta/2} = \mathsf{negl}(\sigma)$, for σ as in Lemma 2. □

D UC Commitment Length Extension Implies OT (cont'd)

Here we present the full proof of Theorem 7. Let Π denote a UC-secure 3κ-bit commitment protocol in the $\mathcal{F}_{\text{MCOM}}[\kappa]$-hybrid model, with a single query.

We make a simplifying assumption that the sender P_s in Π acts as the sender in the underlying κ-bit commitment. (It is easy to see that the proof extends to the other setting too.) In the following we will denote the string committed to by the sender to $\mathcal{F}_{\text{MCOM}}[\kappa]$ as the value q. The input to P_s is a 3κ-bit string m.

Lemma 11 (Informal)**.** *For all 3κ-bit messages m, with probability $1 - \mathsf{negl}(\kappa)$, sender in Π must issue an* reveal *request to $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ (i.e., on value q) in either commit or reveal phase.*

Proof. We prove this by contradiction, by showing a malicious P_r^* which breaks hiding of Π.

Claim 12. *There exists a cheating Π-receiver P_r^* such that for all 3κ-bit message m, if*

$$\Pr[P_s(m) \textit{ doesn't open commitment to } q \textit{ when interacting with honest } P_r \textit{ in } \Pi] \geq \epsilon,$$

then,

$$\Pr[P_r^* \textit{ extracts } m \textit{ when interacting with } P_s(m)] \geq \epsilon - \mathsf{negl}(\kappa).$$

We construct P_r^* as follows: P_r^* runs the extractor for Π to extract m, while "pretending" that q is the all zeroes string. (Note P_r^* doesn't get to see the "short" committed value.)

The reason this works is that if $P_s(m)$ doesn't open the commitment to q, then from the view-point of an honest receiver, it doesn't matter whether $P_s(m)$ commits to q or to all-zeroes. More concretely, consider P_s^* that behaves like $P_s(m)$ in both commit and open phases except it queries all-zeroes instead of q. Then, P_s^* will be able to commit and open to m with probability ϵ when interacting with an honest P_r. This means the extractor must extract m from $P_s^*(m)$, and thus P_r^* extracts m from $P_s(m)$. Therefore, $\epsilon = \mathsf{negl}(\kappa)$ must hold. □

Lemma 13 (Informal)**.** *Suppose that sender in Π opens the commitment to q in either commitment or reveal phase with all but negligible probability (for a random m). Then, there exists a protocol for semi-honest stand-alone OT.*

Proof. We will prove that the following protocol π_{OT} is a semihonest secure protocol for oblivious transfer.

- **Inputs.** P_s acts as the sender in π_{OT} with inputs $(b_0, b_1) \in \{0,1\}^2$. P_r acts as the receiver in π_{OT} with input $\sigma \in \{0,1\}$.
- **Protocol.**
  1. P_s chooses two random strings $m_0, m_1 \in \{0,1\}^{3\kappa}$.
  2. P_s and P_r participate in two executions of the commitment length extension protocol Π, denoted Π_0, Π_1, as follows. For $c = 0, 1$: In execution Π_c:
     - The sender P_s uses Π_c to commit to $m_c \in \{0,1\}^{3\kappa}$. The sender follows instructions of Π_c exactly except whenever Π_c instructs it to make a call to $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ on some value q_c, the sender sends a message ("query", q_c) to P_r.
     - The receiver plays the role of the receiver in Π_c except it ignores messages of the form ("query", $\star$).
       - If $c = \sigma$, then the receiver in π_{OT} runs the simulation extractor for Π_c (which may use the value q_c that P_r received from P_s) to extract m'_c.
       - If $c \neq \sigma$, then the receiver in π_{OT} runs the honest receiver algorithm for Π_c.
  3. P_s chooses two random strings $r_0, r_1 \in \{0,1\}^{3\kappa}$, and sends r_0, r_1 along with values $z_0 = b_0 \oplus \langle m_0, r_0 \rangle$ and $z_1 = b_1 \oplus \langle m_1, r_1 \rangle$.
  4. P_r outputs bit $z = z_\sigma \oplus \langle m'_\sigma, r_\sigma \rangle$.

Correctness of the OT protocol follows from the security of the UC commitment protocol. More concretely, the value extracted by the simulation extractor (interacting with a corrupt sender in the commitment protocol) in execution Π_σ i.e., m'_σ, equals m_σ with all but negligible probability. Given this, it is easy to see that with all but negligible probability P_r's output z equals b_σ.

Next, we prove security of the OT protocol.

Corrupted sender P_s^*. Loosely speaking, by the UC security of Π (against a corrupt sender), corrupt P_s^* cannot distinguish between execution Π_σ (where the receiver plays the role of the honest receiver for Π) and execution $\Pi_{1-\sigma}$ (where the receiver plays the role of the simulation extractor for Π), and so P_s^* has negligible advantage in guessing the receiver input σ. We provide a sketch of the simulation below.

Simulator $\mathcal{S}$ for a corrupt sender in π_{OT} receives (b_0, b_1) as input. To obtain simulated view of execution Π_0, $\mathcal{S}$ chooses random $m_0 \in \{0,1\}^{3\kappa}$ and runs the honest sender algorithm with input m_0 for Π with the honest receiver algorithm for Π, except (as in protocol π_{OT}) sends the value of the sender's query q_0 to $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ to the receiver in the clear. Let v_0 be the resulting view of the sender which in particular includes the query q_0 that $\mathcal{S}$ obtains while acting as $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ to the honest sender.

Similarly, $\mathcal{S}$ chooses a random $m_1 \in \{0,1\}^{3\kappa}$ and runs the honest sender algorithm with input m_1 for Π with the honest receiver algorithm for Π, except (as in protocol π_{OT}) sends the value of the sender's query q_1 to $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ to the receiver in the clear. Let v_1 be the resulting view of the sender which in particular includes the query q_1 that $\mathcal{S}$ obtains while acting as $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ to the honest sender. The rest of the transcript of π_{OT} is generated using the honest sender algorithm for π_{OT} with input (b_0, b_1), and random messages m_0, m_1. Finally, output the view of the honest sender, including the views v_0, v_1.

It is easy to see that the view $v_{1-\sigma}$ is distributed identically to the view generated in protocol $\Pi_{1-\sigma}$ in the real execution. From the UC security of Π (against a corrupt sender), it follows that the view v_σ is indistinguishable from the view generated in protocol Π_σ in the real execution. Given the above, it is easy to see that the output of the simulator is indistinguishable from the output of a corrupt sender P_s^* in the real protocol.

Corrupted receiver P_r^*. We provide a sketch of the simulation, and show its correctness by contradiction.

The simulator $\mathcal{S}$ for a corrupt receiver in π_{OT} receives σ and b_σ as input. To obtain simulated view of execution Π_σ, it begins simulating an execution between the honest sender with random input m_σ and the simulation extractor, in particular $\mathcal{S}$ acts as $\mathcal{F}_{\mathrm{MCOM}}[\kappa]$ and receives the query q_σ from the honest sender which it then forwards to the simulation extractor. Let v_σ be the resulting view of the simulation extractor.

To generate a simulated view of execution $\Pi_{1-\sigma}$, the simulator $\mathcal{S}$ chooses a random $m_1 \in \{0,1\}^{3\kappa}$ and starts an execution of the simulator for a corrupt receiver in Π, denoted $\mathcal{S}'$, with the honest receiver algorithm in Π in *both* the commit and reveal phases, while providing random value $m_{1-\sigma}$ as input of the sender to $\mathcal{S}'$. Note at the end of this simulated interaction, $\mathcal{S}$ obtains the opened query $q_{1-\sigma}$ from $\mathcal{S}'$. Let $v_{1-\sigma}$ be this simulated view of $\Pi_{1-\sigma}$ along with the query $q_{1-\sigma}$. The rest of the transcript of π_{OT} is generated by choosing random values $r_0, r_1 \in \{0,1\}^{3\kappa}$ and setting $z_\sigma = b_\sigma \oplus \langle m_\sigma, r_\sigma \rangle$ and $z_{1-\sigma}$ at random from {0,1}. Finally, output the view of the receiver, including the views v_0, v_1, and values z_0, z_1. We show that the simulated view obtained as described above is indistinguishable from an execution of the real protocol.

It is easy to see that the view v_σ and values q_σ, z_σ are distributed identically to the view generated in protocol Π_σ in the real execution. Also, by the UC security of Π (against a corrupt receiver), the view $v_{1-\sigma}$ and query $q_{1-\sigma}$ are indistinguishable from the view and the query generated in protocol $\Pi_{1-\sigma}$ in the real execution. Given the above, it remains to show that the distribution of $z_{1-\sigma}$ in the simulated view is indistinguishable from the corresponding distribution in the real execution of π_{OT}. Since $z_{1-\sigma}$ is distributed uniformly in the simulated view, it suffices to show that the value $\langle m_{1-\sigma}, r_{1-\sigma} \rangle$ is distributed negligibly close to uniform in the *real* protocol.

To show this we use the following fact. Suppose there exists a distinguisher that guesses $\langle m_{1-\sigma}, r_{1-\sigma} \rangle$

for a random $r_{1-\sigma}$ with $1/\mathrm{poly}(\kappa)$ advantage, then by Goldreich-Levin theorem [20], it can compute $m_{1-\sigma}$ with probability $1/\mathrm{poly}(\kappa)$. We derive a contradiction as follows:

Game 0: This is the real protocol execution of π_{OT}.

Game 1: This game is identical to **Game 0** except we replace honest sender in execution $\Pi_{1-\sigma}$ of protocol π_{OT} by the simulated honest sender (i.e., this is the equivocal simulator who's interacting with a cheating receiver P_r^* that simply runs the honest receiver strategy but also outputs the internal coin tosses) as follows:

- For the protocol messages that are already in $\Pi_{1-\sigma}$, we can use the simulator.
- For the query $q_{1-\sigma}$, we use the simulated opening to $q_{1-\sigma}$. (This is where we use the fact that with all but negligible probability, an honest sender opens to $q_{1-\sigma}$ in either the commit or reveal phase, which means that $q_{1-\sigma}$ will be part of either the commit phase or the open phase transcript, and therefore the simulator must simulate $q_{1-\sigma}$. In particular, if the honest sender opens to $q_{1-\sigma}$ in the open phase, then the simulator knows m when simulating $q_{1-\sigma}$, in which case $q_{1-\sigma}$ could leak κ bits of information about $m_{1-\sigma}$.)

By the UC security of Π (against a corrupt receiver), the distribution of the view $v_{1-\sigma}$ in execution $\Pi_{1-\sigma}$ and the distribution of the query $q_{1-\sigma}$ in **Game 1** are indistinguishable from the corresponding distribution of the view and the query generated in protocol $\Pi_{1-\sigma}$ in **Game 0**. This suffices to show that **Game 1** is indistinguishable from **Game 0**.

In particular, this implies that a distinguisher that guesses $m_{1-\sigma}$ with $(1/\mathrm{poly}(\kappa)) - \mathsf{negl}(\kappa)$ advantage in **Game 0** will be able to do the same in **Game 1** with at least $(1/\mathrm{poly}(\kappa)) - \mathsf{negl}(\kappa)$ advantage.

Game 2: This game is identical to **Game 1** except we use simulated view of π_{OT} as in Game 2, but guess the query $q_{1-\sigma}$ at random (instead of obtaining it via the simulated honest sender).

Clearly, when the guess for query $q_{1-\sigma}$ equals the value produced by the simulator for a corrupt receiver in **Game 1**, the view of the eavesdropper is identical in **Game 1** and **Game 2**. This happens with probability $2^{-\kappa}$ since the guess for $q_{1-\sigma} \in \{0,1\}^\kappa$ is made at random in **Game 2**. Therefore, in this case, the eavesdropper must guess the value of m with probability at least $2^{-\kappa} \cdot ((1/\mathrm{poly}(\kappa)) - \mathsf{negl}(\kappa))$.

However, note that **Game 2** is designed in a way such that the transcript of the OT protocol is statistically independent of $m_{1-\sigma}$. Therefore, any eavesdropper that relies solely on the transcript of **Game 2** must be able to guess the value of a randomly chosen $m_{1-\sigma} \in \{0,1\}^{3\kappa}$ with probability at most $2^{-3\kappa}$.

Thus we have a contradiction. Therefore, we conclude that the value $\langle m_{1-\sigma}, r_{1-\sigma} \rangle$ is distributed negligibly close to uniform in the real OT protocol, and that the simulated view output by simulator for a corrupt receiver in π_{OT} is indistinguishable from the view of a corrupt receiver in the real protcol. □

E Efficient Commitments for Cut-and-Choose

While our rate 1 construction has good concrete efficiency for large string commitments, the case of short string commitments leaves a lot to be desired. An obvious approach to handle short strings is simply to concatenate these strings together to form one large string, and then use the rate 1 construction with this string as the input message. While this approach does provide a concrete rate close to 1 when the number of instances is large, it has the drawback that all instances of short strings must be opened simultaneously. In this section, we design more efficient commitment scheme for handling multiple instances of κ-bit strings with two opening phases (as required in techniques such as cut-and-choose). The extension to three or more opening phases is straightforward.

For $i \in [n]$, let the i-th κ-bit string be denoted by m_i, and let $m = (m_1, \ldots, m_n)$. Let p denote the number of opening phases, and for $j \in [p]$, let u_j denote the characteristic vector of the subset $S_j \subseteq [n]$ of

the strings that need to opened in the j-th opening phase. Note that u_j is not known to the sender during the commit phase.

Our high level idea is as follows. As in our rate 1 construction, we let the sender encode m in to m' using the rate 1 encoding scheme. In addition, for each $i \in [p]$, the sender uses the rate 1/2 encoding scheme (naturally derived from Enc) to encode the zero string $(0, \ldots, 0) \in \mathbb{F}^n$ twice using independent randomness to obtain codewords $z^{(1)}, z^{(2)}$ (each of length $2n'$). Next the sender prepares to send symbols through the Rabin-OT oracle. For this, it constructs $M_k = (m'_k, z_k^{(1)}, z_k^{(2)})$ for $k \leq n'$, and symbols $M_k = (z_k^{(1)}, z_k^{(2)})$ for $k \in \{n'+1, \ldots, 2n'\}$, as the k-th input to the Rabin-OT oracle. Then, it transmits M_k through Rabin-OT oracle with parameter $\delta' = \delta/2$ (where δ is the best parameter for obtaining commitments on strings of length $n\kappa$). Then, in the j-th opening phase, the receiver sends the randomness (alternatively, a seed to a PRG) to encode u_j into u'_j using the rate 1 encoding scheme. Now, denote the underlying polynomials (cf. Figure 4) for (1) the rate 1 encoding of m by q_m, (2) the rate 1/2 encoding of $z^{(j)}$ as $q_z^{(j)}$, and (3) the rate 1 encoding of u_j by q_u^j. In the j-th opening phase, the sender simply reveals the polynomial $q^{(j)} = (q_m \cdot q_u^j) + q_z^{(j)}$. Now, let $\{\tilde{M}_k\}_{k \in J}$ denote the messages received by the receiver. The receiver checks if for all $k \in J \cap [n']$, it holds that $\tilde{M}_k = (\tilde{m}'_k, \tilde{z}_k^{(1)}, \tilde{z}_k^{(2)})$ satisfies $q^{(j)}(k) = (\tilde{m}'_k \cdot q_u^j(k)) + \tilde{z}_k^{(j)}$. If the check succeeds, then the receiver computes $v_i = q^{(j)}(e_i)$, where e_i are the publicly known points as described in Figure 4. If for all $i \notin S_j$, it holds that $v_i = 0$, then receiver outputs $\{v_j\}_{j \in S_j}$ and terminates, else it outputs $\perp$ and terminates. Let c_1, c_2, c_3 represent our concrete cost of realizing commitments on strings of length $n\kappa$ in the offline, the online commit, and the online reveal phases respectively. It can be verified that the cost of the above scheme that implements n instances of κ-bit commitments with two opening phases is $\approx 8c_1, 2c_2, 2c_3$ in the offline, the online commit, and the online reveal phases respectively.