Two-Round Adaptively Secure Multiparty Computation from Standard Assumptions

Fabrice Benhamouda[1(✉)], Huijia Lin[2], Antigoni Polychroniadou[3], and Muthuramakrishnan Venkitasubramaniam[4]

[1] IBM Research, Yorktown Heights, NY, USA
fabrice.benhamouda@normalesup.org
[2] University of California, Santa Barbara, CA, USA
[3] Cornell Tech, New York, NY, USA
[4] University of Rochester, Rochester, NY, USA

Abstract. We present the first *two-round* multiparty computation (MPC) protocols secure against malicious *adaptive* corruption in the common reference string (CRS) model, based on DDH, LWE, or QR. Prior two-round adaptively secure protocols were known only in the two-party setting against semi-honest adversaries, or in the general multiparty setting assuming the existence of indistinguishability obfuscation (iO).

Our protocols are constructed in two steps. First, we construct two-round oblivious transfer (OT) protocols secure against malicious adaptive corruption in the CRS model based on DDH, LWE, or QR. We achieve this by generically transforming any two-round OT that is only secure against static corruption but has certain oblivious sampleability properties, into a two-round adaptively secure OT. Prior constructions were only secure against semi-honest adversaries or based on iO.

Second, building upon recent constructions of two-round MPC from two-round OT in the weaker *static* corruption setting [Garg and Srinivasan, Benhamouda and Lin, Eurocrypt'18] and using equivocal garbled circuits from [Canetti, Poburinnaya and Venkitasubramaniam, STOC'17], we show how to construct two-round *adaptively* secure MPC from two-round *adaptively* secure OT and constant-round *adaptively* secure MPC, with respect to both malicious and semi-honest adversaries. As a corollary, we also obtain the first 2-round MPC secure against semi-honest adaptive corruption in the plain model based on augmented non-committing encryption (NCE), which can be based on a variety of assumptions, CDH, RSA, DDH, LWE, or factoring Blum integers. Finally, we mention that our OT and MPC protocols in the CRS model are, in fact, adaptively secure in the Universal Composability framework.

1 Introduction

The notion of secure multi-party computation (MPC) allows N mutually distrustful parties $P_1, \ldots, P_N$ to securely compute a functionality $f(\bar{x}) =$

A. Beimel and S. Dziembowski (Eds.): TCC 2018, LNCS 11239, pp. 175–205, 2018.
https://doi.org/10.1007/978-3-030-03807-6_7

$f_1(\bar{x}), \ldots, f_N(\bar{x})$ of their corresponding private inputs $\bar{x} = x_1, \ldots, x_N$, such that party P_i receives the value $f_i(\bar{x})$. Loosely speaking, the security requirements are that the parties learn nothing more from the protocol than their prescribed output, and that the output of each party is distributed according to the prescribed functionality. This should hold even in the case that a malicious adversary seizes control of an arbitrary subset of the parties and make them arbitrarily deviate from the protocol. A major achievement in the 80's is demonstrating that any function that can be efficiently computed, can be efficiently computed securely [3,29,38]. Since then, the round complexity of computing general functionalities has been a central question in the area of MPC.

Answering this question depends on what powers the adversaries have. In the *static corruption* model, the adversary may seize control, or *corrupt*, a subset of parties *before* the protocol begins, and dictate their behavior throughout the protocol execution. A stronger and more realistic model is the *adaptive corruption* model, where the adversary can decide to corrupt more parties at any time *during* the execution of the protocol. The adaptive corruption model captures "hacking attacks" where an adversary has the capability to seize control of parties' machines at any time, through for instance known vulnerabilities or backdoor; in an extreme, the adversary may eventually corrupt all parties. Protecting against such attacks provides stronger security guarantees. Moreover, security against adaptive corruption is instrumental for achieving everlasting security and leakage resilience.

In the static corruption model, a long line of research on two-round protocols [2,6–9,14,20,25–27,31,34,36] that culminated in two recent works by Benhamouda and Lin [5] and Garg and Srinivasan [28] has completely resolved the round complexity of MPC from minimal assumptions. The works of [5] and [28] constructed two-round MPC protocols from any two-round oblivious transfer protocols, in the Common Reference String (CRS) model.[1] Moreover, in the semi-honest setting, these works provide two-round protocols in the plain model (i.e., without CRS).

In contrast, the round-complexity of MPC in the adaptive corruption model is far from being resolved. Prior works [14,16,22,26] constructed 2-round MPC protocols secure against adaptive corruption, based on the strong assumption of Indistinguishability Obfuscation (iO) for polynomial-sized circuits, and other standard assumptions. However, the security of current indistinguishability obfuscation schemes is not well understood. When restricting to using only standard assumptions, Damgård et al. [23] construct 3-round protocols based on LWE for all-but-one corruptions and Canetti et al. [17] construct a constant round protocol based on simulatable public key encryption for arbitrary corruptions. Only in the most restricted case of 2-Party Computation (2PC) in the presence of semi-honest adversaries, they gave a *two-round* protocol based on

[1] Actually, the protocol of [5] additionally relies on Non-Interactive Zero-Knowledge (NIZK) proofs in the CRS model. But as observed in [28] and this work, the use of NIZK can be removed.

the minimal assumptions. The state-of-affairs leaves the following basic questions open:

Can we have the following based on standard assumptions?
- *Two-round MPC secure against adaptive corruption by semi-honest adversaries in the plain model.*
- *Two-round MPC secure against adaptive corruption by malicious adversaries in the CRS model.*

In fact, the second question remains open even for the special case of 2PC protocols computing the Oblivious Transfer (OT) functionality. In the literature, there are 2-round OT protocols secure against either static corruption by malicious adversaries, *or* adaptive corruption by semi-honest adversaries, from various assumptions [18,19,37]. However, when considering adaptive corruption by malicious adversaries, the best protocols based on standard assumptions have 3 rounds whether assuming erasures [19] or not [1].[2] Therefore, another basic question that remains open so far is,

Can we achieve a two-round OT protocol that is secure against adaptive corruption by malicious adversaries in the CRS model from standard assumptions?

In this work, we answer all above questions affirmatively, obtaining two-round MPC protocols secure against adaptive corruption by semi-honest adversaries in the plain model from minimal assumptions, and 2-round protocols secure against malicious adversaries in the CRS model from any of the following assumptions: Decisional Diffie-Hellman (DDH), Quadratic Residuosity (QR), or Learning with Error (LWE). Our constructions satisfy the stronger UC-security notion [10].

Our Results

We present our results in the *local* CRS model where every session of protocol execution has a local independently sampled CRS. We believe that our protocol constructions and security proofs can be easily adapted to the *single* CRS model where all sessions share a single CRS as in [15]; see Sect. 2.4 for more discussion.

Towards constructing 2-round MPC protocols secure against adaptive corruption, or adaptive-MPC for short, we first show that this task can be reduced to constructing a 2-round OT protocol secure against adaptive corruption, or adaptive-OT for short, at the presence of either semi-honest or malicious adversaries. More precisely,

Theorem 1.1 (Informal). *Assuming the existence of a two-round oblivious transfer protocol and a constant-round MPC protocol secure against adaptive*

[2] Abdalla et al. [1] constructed a two-round OT protocol secure against the weaker semi-adaptive corruption model where the adversary corrupts one of the two parties at the beginning of the execution and the other party adaptively during or after the execution. It is known that such a protocol can be generically converted to become secure against adaptive corruption using NCE. However, the resulting protocol would be 3-round.

corruption by malicious adversaries in the CRS model (resp. semi-honest adversaries in the plain model), there exists a 2-round MPC protocol for any functionality f that is UC-secure against adaptive corruption of any subset of the parties by malicious adversaries, in the CRS model (resp. or semi-honest adversaries in the plain model).

At a high-level, our construction follows the blueprint of the recent constructions of 2-round MPC from 2-round OT in the static corruption model [5,28]. Their key idea is collapsing the number of rounds of arbitrary multi-round MPC protocols into just 2, using just garbled circuits and 2-round OT. Following the same technique, we show that adaptive security follows naturally, when the underlying garbled circuits and OT are also adaptively secure. The work by Canetti et al. [17] constructs adaptively secure garbled circuits, called *equivocal garbling scheme*, from the minimal assumption of one-way functions. However, using equivocal garbled circuits directly only allows us to collape rounds of *constant-round* MPC protocols; otherwise, the resulting 2-round protocol would become inefficient (see Sect. 2.3 for more discussion). The work of Canetti et al. [17] also constructs constant-round adaptive-MPC protocols based on simulatable PKE. Thus, it boils down to construct 2-round adaptive-OT.

When the adversaries are semi-honest, two-round adaptive-OT can be based on augmented Non-Committing Encryption (NCE) [15], which in turn can be based on CDH, RSA, DDH, LWE or factoring Blum-integers [18]. Furthermore, constant-round MPC secure against adaptive corruption can be be constructed from NCE (which is implied by augmented NCE) and semi-honest two-round adaptive-OT [17]. Thus, we obtain the following corollary.

Corollary 1.2. *Assuming augmented non-committing encryption, there exists a 2-round MPC protocol for any functionality f that is UC-secure against adaptive corruption of any subset of the parties by semi-honest adversaries.*

However, there are no known constructions of 2-round adaptive OT protocols against malicious adversaries even in the (local) CRS model. The natural approach of using non-interactive zero-knowledge proofs to convert a *semi-honest* adaptive-OT protocol, say the one in [15], into a *malicious* adaptive-OT protocol require additional rounds, specifically, to incorporate a coin-tossing protocol. The work of [1] comes close by achieving a weaker notion of semi-adaptivity in two rounds based on DDH. Our main technical contribution is constructing two-round adaptive-OT against malicious adversaries from various assumptions. More precisely,

Theorem 1.3 (Informal). *Assuming DDH, QR, or LWE, there exists a 2-round OT protocol that is UC-secure against adaptive corruption by malicious adversaries in the CRS model.*

Combined with previous theorem and the construction of constant-round adaptive-MPC in [17] (which can be constructed from simulatable PKE, which can itself be build from DDH, QR, or LWE too [18]) we obtain as a corollary 2-round adaptive-MPC against malicious adversaries from the same assumptions:

Corollary 1.4. *Assuming DDH, QR, or LWE, there exists a 2-round MPC protocol for any functionality f that is UC-secure against adaptive corruption of any subset of the parties by malicious adversaries in the CRS model.*

To achieve the above theorem, we provide a generic framework that compiles any OT protocol secure against static corruption, or static-OT for short, with appropriate "oblivious sampleability" properties, to a full-fledged adaptive-OT protocol, in just 2-rounds. Roughly speaking, oblivious sampleability refers to the following properties: *(i) Receiver-oblivious-sampleability*:one can obliviously sample the OT receiver's message, and claim that an honestly generated receiver's message *was* obliviously sampled, and similarly *(ii) Sender oblivious sampleability*: one can obliviously sample the sender's message, and claim that an honestly generated sender's message for *random* input strings was obliviously sampled. Then, we show that static-OT with such oblivious sampleability can be instantiated from various concrete assumptions, including DDH, or LWE, or QR.

2 Technical Overview

We start with an overview of our construction of 2-round adaptive-OT and then move to 2-round adaptive MPC in the local CRS model. In the end, we briefly discuss future work on extending our results to the single CRS model.

2.1 2-Round Adaptive-OT

To construct a 2-round adaptive-OT Π_3, we start with a basic 2-round static-OT[3] Π with the special property of sender and receiver *oblivious sampleability*, and transform it in three steps to gradually achieve security against different adaptive corruption scenarios:

- *Sender semi-adaptive corruption* refers to the case where the receiver is corrupted at the beginning of the protocol execution and the sender is corrupted after the execution, i.e. post-execution.
- *Receiver semi-adaptive corruption* refers to the symmetric case where the sender is corrupted from the beginning and the receiver is corrupted post-execution.
- *Semi-adaptive corruption* refers to either of the above two scenarios. In comparison, full fledged *adaptive corruption* considers the additional scenario where neither sender nor receiver is corrupted during execution, and both corrupted post-execution in an arbitrary order; we refer to the latter *semi-honest post-execution corruption*.

Starting with a static-OT Π with sender and receiver oblivious sampleability,

[3] In fact, it suffices if the OT protocol Π is secure against semi-honest senders, and malicious receivers.

- *In Transformation 1,* we transform Π into Π_1 that achieves security against sender-semi-adaptive corruption (and preserves security in static corruption scenarios). This step crucially relies on the sender oblivious sampleability of Π, and preserves receiver oblivious sampleability.
- *In Transformation 2,* we rely on receiver oblivious sampleability to transform Π_1 into Π_2 to achieve security under receiver-semi-adaptive corruption, while preserving security under sender-semi-adaptive corruption. Π_2 is now secure under semi-adaptive corruption.
- *In Transformation 3,* finally, we transform the semi-adaptive-OT Π_2 into an adaptive-OT Π_3, using additionally augmented NCE.

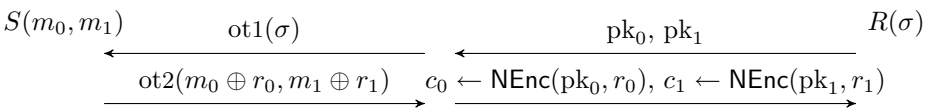
Below, we describe ideas in these three transformation, starting with the third transformation.

Transformation 3: Semi-Adaptive-OT to Adaptive-OT. Consider a semi-adaptive OT Π_2, whose algorithms for generating the CRS, the sender, and receiver messages are denoted as Setup, S_2, R_2. The only corruption scenario that Π_2 does not handle is *semi-honest post-execution corruption* (i.e., neither sender nor receiver is corrupted during execution, but both corrupted post-execution in an arbitrary order). It is known that semi-adaptive OT can be transformed into adaptive-OT by sending messages of the former using private channels implemented by Non-Committing Encryption (NCE) [24], which, however, produces a three-round protocol. In two rounds, the above corruption case *alone* can be handled using *augmented* NCE as done in the construction of semi-honest adaptive-OT by Canetti, Lindell, Ostrovsky, and Sahai (CLOS) [15]. Below, we use their protocol to lift the security of Π_2.

Augmented NCE. NCE is a public key encryption with the special property of *equivocality*: one can simulate a pair of pubic key and ciphertext (pk, c) and later "open" them to any plaintext m, by efficiently finding randomness ρ and τ that "explains" the public key and ciphertext consistently w.r.t. m (meaning $\tilde{c} = \mathsf{NEnc}(\tilde{\mathrm{pk}}, m; \rho)$, $(\tilde{\mathrm{pk}}, \tilde{\mathrm{sk}}) = \mathsf{NGen}(1^\lambda; \tau)$, $m = \mathsf{NDec}(\tilde{\mathrm{sk}}, \tilde{c})$).

A NCE is "augmented" if it has *i) oblivious key sampleability*: one can obliviously sample a public key pk' without knowing any corresponding secret key, and *ii) inverse key sampleability*: one can claim that an honestly generated or simulated public key pk was sampled obliviously by efficiently finding randomness that would make the oblivious key sampling algorithm produce pk.

"Patch" Semi-Adaptive-OT Π_2 Using Augmented NCE. To handle semi-honest post-execution corruption, we run Π_2 with the CLOS semi-honest adaptive-OT from augmented NCE in parallel as depicted below. The instance of Π_2 is generated using the receiver's choice bit σ and the sender's messages padded with two random strings $(m_0 \oplus r_0, m_1 \oplus r_1)$. In the instance of CLOS, the receiver samples one public key pk_σ honestly with sk_σ, and another $\mathrm{pk}_{1-\sigma}$ obliviously, followed by the sender encrypting the two random pads r_0, r_1 using respectively these two keys.



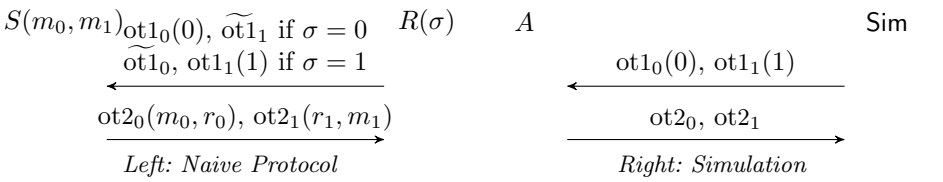
To handle semi-honest post-execution, the trick is simulating the instance $(\mathrm{ot1}, \mathrm{ot2})$ of Π_2 using its simulator Sim_2 *for the case where the sender is statically corrupted* (recall that Π_2 is secure under semi-adaptive corruption). This can be done as the simulator can generate ot2 honestly using just random messages r'_0, r'_1: effectively, the sender of Π_2 is "corrupted" and instructed to act honestly with input r'_0, r'_1. Thus, Sim_2 can be used to simulate and equivocate the receiver's messages. The keys and ciphertexts in the instance of CLOS is simulated relying on properties of augmented NCE. Later, for instance, when the sender is corrupted first post-execution, the simulator, learning (m_0, m_1), finds the right "pads" $r_0 = m_0 \oplus r'_0$, $r_1 = m_1 \oplus r'_1$, and uses the equivocality of NCE to "explain" that the keys and ciphertexts are consistent with r_0, r_1. In the other case, when the receiver is corrupted first post-execution, the simulator, learning σ, m_σ, can explain pk_σ consistently with r_σ, and claim that $\mathrm{pk}_{1-\sigma}$ were obliviously sampled using the inverse oblivious sampleability property.

It might seem that the above transformation can use any semi-honest adaptive OT. This is indeed the case only if semi-honest post-execution corruption was concerned. But, we also want the transformation to preserve security under semi-adaptive corruption (when Π_2 has this property). For that, we rely on special properties of the CLOS protocol; in particular, it is already secure against *malicious* sender, and simulated public keys from the receiver can be easily equivocated. See Sect. 5.5 for more details.

Transformation 2: Handling Receiver-Semi-Adaptive Corruption. We now move to handling the first scenario in semi-adaptive corruption, i.e. receiver semi-adaptive corruption. When the sender is maliciously corrupted from the beginning and the receiver is corrupted post-execution, the simulator needs to *(i)* simulate the receiver's message $\widetilde{\mathrm{ot1}}$ without knowing the choice bit, *(ii)* extract both sender's messages m_0, m_1, *(iii)* and later equivocate $\widetilde{\mathrm{ot1}}$ to any choice bit σ. A common approach in the literature for enabling equivocation is relying on appropriate oblivious sampleability property. We follow this approach and formalize the following receiver oblivious sampleability property.

> *Receiver Oblivious Sampleability:* A two-round OT protocol (in the CRS model) has receiver oblivious sampleability if *(1)* one can obliviously sample receiver's message $\widetilde{\mathrm{ot1}}$, and *(2)* can claim that an honestly generated receiver's message ot1 for any choice bit σ was obliviously sampled, by efficiently finding consistent randomness ρ that would make the oblivious sampling algorithm produce ot1. Furthermore, messages and randomness produced in these two ways are indistinguishable
>
> $$(\mathrm{crs}, \widetilde{\mathrm{ot1}}, \widetilde{\rho}) \approx (\mathrm{crs}, \mathrm{ot1}, \rho).$$

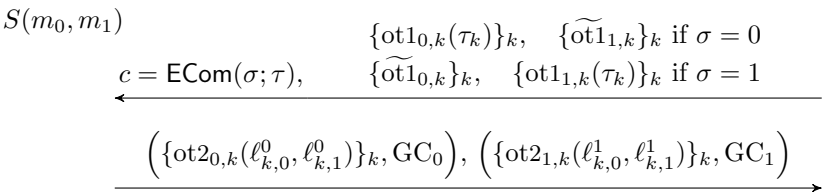
A Naive Idea and its Problem. Given Π_1 with receiver oblivious sampleability, the basic idea is to let the receiver of Π_2 send two messages, where $\mathrm{ot1}_\sigma$ is generated honestly using the choice bit σ while $\widetilde{\mathrm{ot1}}_{1-\sigma}$ is sampled obliviously. The sender then replies $\mathrm{ot2}_0, \mathrm{ot2}_1$ respectively, where $\mathrm{ot2}_b$ is generated honestly w.r.t. $\mathrm{ot1}_b$ using message m_b at slot b and random message r_b at slot $1-b$. See the depiction below on the left.



For the above protocol, simulation under receiver-semi-adaptive corruption can be done as follows: *(i)* the simulator can "plant" honestly generated receiver's messages $\mathrm{ot1}_0, \mathrm{ot1}_1$ for both choice bit 0 and 1. *(ii)* Upon receiving sender's messages $\mathrm{ot2}_0, \mathrm{ot2}_1$, it uses the OT output strings as the extracted sender's messages. Finally, *(iii)* the simulator equivocates the receiver's messages w.r.t. a choice bit by revealing the randomness ρ_σ used for generating $\mathrm{ot1}_\sigma$ honestly, and reverse sampling randomness $\widetilde{\rho}_{1-\sigma}$ for claiming that $\mathrm{ot1}_{1-\sigma}$ were obliviously sampled.

Though receiver-semi-adaptive corruption is resolved, unfortunately, the above protocol is not secure against malicious receivers (even though Π_1 is), as a cheating receiver can use the same strategy the simulator uses and violate sender's privacy.

Fixing the Problem. To overcome this, the simulator needs to have some unique advantage that malicious receivers do not have. Our idea is using an equivocal commitment ECom (in CRS model). More specifically, the receiver should send a ECom commitment c to its choice bit σ, so that, only the simulator can generate a simulated commitment $\tilde{c}$ and open it to both 0 and 1, but not cheating receivers. To incorporate this, the rest of the protocol needs to be modified accordingly: The two instances of OT Π_1 are replaced with two instances of 2 Party Computation (2PC): the b'th instance reveals to the receiver the message m_b conditioned on the receiver having a valid opening of c to b.



Simulation in the receiver-semi-adaptive corruption scenario uses similar ideas as described above w.r.t. the naive protocol. Again, the simulator "plants" valid receiver's messages for both choice bit 0 and 1. In particular, it generates a

simulated ECom commitment $\tilde{c}$ and two sets of ot1 messages corresponding to both opening τ^0, τ^1 of $\tilde{c}$ to 0 and 1, $\{\mathrm{ot1}_{0,k}(\tau_k^0)\}_k$, $\{\mathrm{ot1}_{1,k}(\tau_k^1)\}_k$. To equivocate to any choice bit σ, the simulator can again reveal the randomness used for generating the set $\{\mathrm{ot1}_{\sigma,k}\}$ of messages corresponding to τ_σ, and claim that the other set $\{\mathrm{ot1}_{1-\sigma,k}\}$ was sampled obliviously. The advantage of this protocol is that now malicious receiver cannot copy the simulator's strategy, as it cannot find opening of a ECom commitment to both 0 and 1.

Preserving Security under Sender-Semi-Adaptive Corruption. Furthermore, we show that if Π_1 is secure under sender-semi-adaptive corruption (i.e. where the receiver is maliciously corrupted from the beginning and the sender is corrupted post-execution), the above transformation preserves it. To this end, we need the second message of 2PC to be equivocal. This can be achieved by implementing 2PC using Π_1 and *equivocal garbled circuits* constructed by [17] from one-way functions.

In summary, our transformation 2 produces a semi-adaptive OT, starting from one that is only secure under sender-semi-adaptive corruption (and static corruption of the sender by a semi-honest adversary). We remark that our transformation is quite similar to the transformation presented in the recent work of [28] for achieving some equivocal property of receiver's message. However, their notion of equivocality is tailored for simulation in static corruption cases, and only need to provide *partial* randomness consistent with a choice bit σ. In adaptive corruption, equivocation requires providing *complete* randomness for generating the receiver's message. Thus, the two transformation differ in details; in particular, we crucially rely on receiver oblivious sampleability which is not needed in [28].

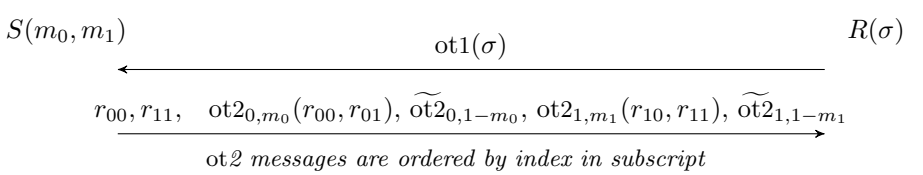
Transformation 1: Handling Sender-Semi-Adaptive Corruption. When the receiver is maliciously corrupted from the beginning and the sender is corrupted post-execution, the simulator needs to *(i)* extract the choice bit σ from the receiver's message ot1, and then obtain the output message m_σ, *(ii)* next simulate the sender's message $\widetilde{\mathrm{ot2}}$ knowing only m_σ, *(iii)* and finally be able to equivocate $\widetilde{\mathrm{ot2}}$ w.r.t. arbitrary $m_{1-\sigma}$. To enable equivocation, we again formulate an oblivious sampleability property now w.r.t. sender's messages.

> *Sender Oblivious Sampleability (Overly Simplified):* Roughly speaking, we want the property that *(1)* one can obliviously sample a sender's message $\widetilde{\mathrm{ot2}}$ (w.r.t. a crs and receiver's message ot1), and *(2)* can claim that an honestly generated sender's message ot2 for *random* messages r_0, r_1 was obliviously sampled, by efficiently finding randomness ρ that would make the oblivious sampling algorithm output ot2. Moreover, messages and randomness generated in these two ways are indistinguishable:
>
> $$(\mathrm{crs}, \mathrm{ot1}, \widetilde{\mathrm{ot2}}, \widetilde{\rho}) \approx (\mathrm{crs}, \mathrm{ot1}, \mathrm{ot2}, \rho).$$

We remark that unfortunately the above description is overly simplified; for the proof to go through, the actual sender oblivious sampleability is more complex.

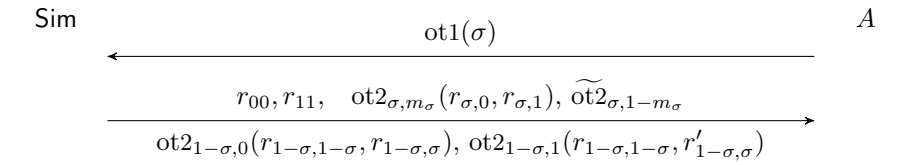
However, for simplicity of exposition, we use the above simple version in this high-level overview.

Staring from a static-OT Π with sender oblivious sampleability, we construct a bit-OT Π_1 with security under sender semi-adaptive corruption. (Note that constructing bit-OT is without loss of generality, as it implies string-OT with the same security in different corruption scenarios.) The basic idea is again to let sender of Π_1 send multiple messages of Π. This redundancy allows simulation to "plant" honestly generated sender's messages for both input bit 0 and 1, at either slot. Then, later to equivocate to $m_{1-\sigma}$, the simulator can correctly open the message generated with value $m_{1-\sigma}$, and claim that the other message was sampled obliviously.



More specifically, Π_1 (depicted above) works as follows. Upon receiving a single receiver's message ot1 of Π, the sender replies two pairs of sender's messages of Π: The b'th pair contains $\mathrm{ot2}_{b,m_b}, \widetilde{\mathrm{ot2}}_{b,1-m_b}$, where the former is honestly generated for random messages (r_{b0}, r_{b1}), and the latter obliviously sampled. The 4 ot2 messages are ordered according to their index. In addition, the sender reveals in the clear r_{00} and r_{11}. It is easy to see that an honest receiver with a choice bit σ will recover exactly the string $r_{\sigma\sigma}$ from message $\mathrm{ot2}_{\sigma,m_\sigma}$, and from the order of $\mathrm{ot2}_{\sigma,m_\sigma}$ in the 4 ot2 messages, it learns m_σ.

Sender Semi-Adaptive Corruption. Simulation in the sender-semi-adaptive corruption scenario can now be achieved as follows. *(i)* The simulator can extract the receiver's choice bit σ using the simulator of Π for the case with a (statically corrupted) malicious receiver, and then learns the output string m_σ. *(ii)* To simulate sender's message, it generates the σ'th pair $(\mathrm{ot2}_{\sigma,m_\sigma}, \widetilde{\mathrm{ot2}}_{\sigma,1-m_\sigma})$ honestly as Π_1 specifies, and simulates the $1-\sigma$'th pair by generating both $\mathrm{ot2}_{1-\sigma,0}, \mathrm{ot2}_{1-\sigma,1}$ honestly, using the same message $r_{1-\sigma,1-\sigma}$ at slot $1-\sigma$ and *different* random strings at slot σ; in addition r_{00}, r_{11} are revealed in the clear.



Both input messages to ot*2, and* ot*2 messages themselves are ordered by index.*

(iii) Finally, to equivocate to sender's true inputs (m_0, m_1), the simulator can reveal the randomness used for generating the σ'th pair $(\mathrm{ot2}_{\sigma,m_\sigma}, \widetilde{\mathrm{ot2}}_{\sigma,1-m_\sigma})$, which were generated correctly using m_σ. For the $1-\sigma$'th pair $\mathrm{ot2}_{1-\sigma,0}, \mathrm{ot2}_{1-\sigma,1}$,

the simulator needs to "explain" w.r.t. $m_{1-\sigma}$. This can simply be done by revealing the randomness used for generating $\mathrm{ot2}_{1-\sigma,m_{1-\sigma}}$ honestly, and claim that $\mathrm{ot2}_{1-\sigma,1-m_{1-\sigma}}$ were sampled obliviously.

Making the above idea work turns out to require a more complex formulation of the sender oblivious sampleability property. Roughly speaking, the complexity stems from the fact that when reducing to sender oblivious sampleability, to simulate the adversary's view, the reduction needs to obtain the choice bit σ of the corrupted receiver (as simulation of sender's message depends on σ). This means sender oblivious sampleability needs to hold against adversaries (the reduction) who receive help in "breaking" a receiver's message of its choice. We omit the complexity here and refer the reader to Sect. 5.3 for more detail.

Fortunately, we can achieve such strong sender oblivious sampleability, as well as receiver oblivious sampleability, from various concrete assumptions, including DDH, QR, and LWE.

2.2 Instantiation of Static-OT with Oblivious Sampleability

We now briefly summarize ideas behind our instantiation from concrete assumptions. To construct the static-OT with oblivious sampleability, we start from a variant of the OT construction based on Smooth Projective Hash Functions (SPHFs) from Halevi and Kalai [32] which generalizes the construction from Naor and Pinkas [35]. In our setting, the SPHF we consider is a primitive which allows some party to generate a hash value H of a pair (ct, σ') of a ciphertext ct and a value σ', together with what is called a projection key hp so that: if ct is indeed a ciphertext of σ', it is possible to compute H from hp and the random coins used to generated ct. But if ct is not a ciphertext of σ', H looks completely random even knowing hp.

The construction works as follows. The CRS contains a public key of an encryption scheme. The receiver's message is a ciphertext ct of the selection bit σ. The sender then uses the SPHF to mask its inputs x_0 and x_1, so that only the one corresponding to the plaintext of ct can be unmasked. More precisely, the sender's message consists of two projection keys hp_0 and hp_1 for the ciphertext ct and the values 0 and 1 respectively, as well as the values $\mathsf{H}_0 \oplus x_0$ and $\mathsf{H}_1 \oplus x_1$ where H_0 and H_1 are the two hash values corresponding to hp_0 and hp_1. Using the random coins used to generate ct, the receiver, can recover H_σ and then x_σ. But the value $x_{1-\sigma}$ will remain completely hidden, masked by $\mathsf{H}_{1-\sigma}$ which looks random to the receiver.

To achieve oblivious sampleability, we just need ciphertexts of the encryption scheme and projection keys of the SPHF to be obliviously sampleable. We can instantiate them using the ElGamal encryption scheme and the associated SPHF from [21], which already satisfies the oblivious sampleability requirements. This directly gives a static-OT with oblivious sampleability under the Decisional Diffie-Hellman (DDH) assumption.

To instantiate the scheme under the Quadratic Residuosity assumption (QR), we start from the Goldwasser-Micali [30] encryption scheme and the SPHF from [21]. While the Goldwasser-Micali encryption scheme satisfies ciphertext

oblivious sampleability, we do not know how to obliviously sample the projection keys of the associated SPHF. The issue is that projection keys are quadratic residues which we do not know how to sample obliviously. To solve this issue, we slightly change the SPHF to use the group of signed quadratic residues instead [33].

Finally, we show how to achieve a slightly weaker variant of 2-round static-OT, called *half-OT*, with oblivious sampleability under LWE. Roughly speaking, in a half-OT, the sender has a bit b and a single message m, and the receiver with choice bit σ only receives m if $b = \sigma$. We show that this weaker variant of *half-OT*, is already sufficient for our transformation to obtain adaptive-OT. We then instantiate half-OT essentially using the IND-CPA encryption scheme and the SPHF from [4] based on LWE. At a very high-level, the encryption scheme can be seen as the dual-Regev encryption scheme where decryption is done using a full trapdoor for the lattice, to ensure that incorrect ciphertexts are far away from the lattice in all directions (otherwise, we do not know how to prove smoothness of the associated SPHF).

Please see Sect. 5.6 for more details of our instantiation.

2.3 From Adaptive-OT to Adaptive-MPC

Two recent works [5,28] constructed 2-round static-MPC in the CRS model from 2-round static-OT. Actually, the protocol presented in [5] additionally relies on NIZK; but as implicitly observed in [28] and in this paper the use of NIZK can be removed. Moving to the adaptive setting, the natural approach is replacing static-OT with adaptive-OT and ask whether the resulting MPC protocols become adaptively secure. We give affirmative answer. At a very high-level, the proof follows similar ideas as in [5,28]. Still, the formal proof requires carefully examination of all adaptive corruption scenarios and new analysis. In particular, the garbled circuits used in the protocols need to be equivocal for adaptive security to hold.

A subtle issue arises when using equivocal garble circuits: If using them modularly as black-box, we can only collapse rounds of *constant*-round MPC protocols, as opposed to any polynomial-round protocols as in [5,28]. The overall approach of [5,28], followed by this work, is using garbled circuits and OT to collapse rounds of a multi-round MPC protocol. The resulting protocol generates chains of garbled circuits, where each circuit in a chain corresponds to one round in the original MPC protocol, has the lables of the next garble circuit in the chain hardcoded inside. Equivocating a chain entails recursively equivocating the garbled circuits in it. Due to the complexity requirement of equivocal garbling scheme, the size of the equivocal garbled circuits grows exponentially with the length of the chain. As a result, we can only collapse rounds of constant-round MPC protocols. (Note that this issue does not exist in the static setting, simulating a chain of standard garbled circuits does not lead to exponential size-growth.) We can alternatively address the issue of exponential size-growth by applying the techniques of [17] for constructing equivocal garbling scheme

(instead of using equivocal garbled circuits as black-boxe).[4] For simplicity and modularity, we take the first approach and collapse rounds of the constant-round MPC protocols from [17] using the equivocal garbling scheme in the same work. See Sect. 6 for the new protocol and analysis.

In terms of writing, we follow the protocol of [5], but for convenience, we present directly the entire protocol without using their intermediate abstraction (namely witness selectors and garbled interactive circuits); this avoids re-defining every intermediate notion in the adaptive setting, which would add unnecessary complexity.

2.4 Future Work: Moving to the Single CRS Model

Our constructions are in the local CRS model, where every session of protocol execution has its "local" independently sampled CRS. A more stringent model is the single CRS model as formalized in [15], where all sessions share a single CRS.[5] We believe that our construction of 2-round adaptive-OT can be adapted to the single CRS model, and when plugging such an OT in our construction of MPC, the resulting 2-round adaptive MPC protocols also work with single CRS. Recall that we gradually transform a static-OT with sender and receiver oblivious sampleability into an adaptive-OT in three steps. We believe that these transformation also works in the single CRS model. Thus it boils down to instantiate static-OT with oblivious sampleability in the single CRS model from concrete assumptions. The main difference from our current instantiation in the local CRS model is that in the single CRS model, the protocols must satisfy certain non-malleability or simulation-extractability property. But, they can be easily achieved using CCA encryption, which is implied by DDH, QR, and LWE. We leave the formal proof as future work.

3 Preliminaries

3.1 Notation

Throughout the paper $\lambda \in \mathbb{N}$ will denote the security parameter. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if $\forall c \; \exists \; n_c$ such that if $n > n_c$ then $f(n) < n^{-c}$. We will use $\text{negl}(\cdot)$ to denote an unspecified negligible function. We often use $[n]$ to denote the set $\{1, ..., n\}$. The concatenation of a with b is denoted by $a||b$. Moreover, we use $d \leftarrow \mathcal{D}$ to denote the process of sampling d from the

[4] For example, the complexity of all equivocal garbled circuits can simply be proportional to the entropy of the secrets that need to be equivocated, which in the case of MPC are the inputs and randomness of the uncorrupted parties.

[5] We emphasize that the CLOS model of single CRS should be differentiated from the global CRS model formalized in [11]. The key difference lies in that the latter allows the environment to access the global CRS (and hence the CRS cannot be programmed), whereas in the former all protocol execution can access the same CRS but not the environment.

distribution $\mathcal{D}$ or, if $\mathcal{D}$ is a set, a uniform choice from it. If $\mathcal{D}_1$ and $\mathcal{D}_2$ are two distributions, then we denote that they are statistically close by $\mathcal{D}_1 \approx_s \mathcal{D}_2$; we denote that they are computationally indistinguishable by $\mathcal{D}_1 \approx_c \mathcal{D}_2$; and we denote that they are identical by $\mathcal{D}_1 \equiv \mathcal{D}_2$.

For the sake of simplicity, we suppose that all circuits in a circuit class have the same input and output lengths. This can be achieved without loss of generality using appropriate padding. We recall that for any T-size circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a universal poly(T)-size circuit family $\{U_\lambda\}_{\lambda\in\mathbb{N}}$ such that for any $\lambda \in \mathbb{N}$, any circuit $C \in \mathcal{C}_\lambda$ with input and output lengths n, l, and any input $x \in \{0,1\}^n$, $U_\lambda(C, x) = C(x)$.

3.2 Equivocal Garbling Scheme

Definition 3.1 (Equivocal Garbling Scheme [17]). Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a poly-size circuit class with input and output lengths n and l. A *garbled circuit* scheme GC for $\mathcal{C}$ is a tuple of four polynomial-time algorithms $\mathsf{GC} = (\mathsf{GC.Gen}, \mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$:

Input Labels Generation: $\mathbf{keys} \leftarrow \mathsf{GC.Gen}(1^\lambda)$ generates input labels $\mathbf{keys} = \{\mathsf{keys}_i\}_{i\in[n]}$ (with $\mathsf{keys}_i[b] \in \{0,1\}^\kappa$ being the input label corresponding to the value b of the i-th input wire) for the security parameter λ, input length n, and input label length κ;

Circuit Garbling: $\widehat{C} \leftarrow \mathsf{GC.Garble}(\mathbf{keys}, C; \sigma)$ garbles the circuit $C \in \mathcal{C}_\lambda$ into $\widehat{C}$;

Evaluation: $y = \mathsf{GC.Eval}(\widehat{C}, \{\mathsf{keys}_i[x_i]\}_{i\in[n]})$ evaluates the garbled circuit $\widehat{C}$ using input labels $\mathsf{keys}_i[x_i]$ for input some input $x = (x_1, \ldots, x_n)$ and returns the output $y \in \{0,1\}^l$;

Simulation: $(\widetilde{\mathsf{keys}}, \widetilde{C}, \mathsf{st}) \leftarrow \mathsf{GC.Sim}(1^\lambda, y)$ simulates input labels $\widetilde{\mathsf{keys}}$, a garbled circuit $\widetilde{C}$ and state st for the security parameter λ on the output $y \in \{0,1\}^l$;

Equivocation: $(\mathsf{keys}', \sigma) \leftarrow \mathsf{GC.Equiv}(C, x, \mathsf{st})$ such that given C and x, the simulator generates (inactive) labels and fake randomness σ of the garbling that makes $\widetilde{C}, \mathsf{keys}'$ look like a real garbling of C, x.

satisfying the following security properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any circuit $C \in \mathcal{C}_\lambda$, for any input $x \in \{0,1\}^n$, for any $\mathbf{keys}$ in the image of $\mathsf{GC.Gen}(1^\lambda)$ and any $\widehat{C}$ in the image of $\mathsf{GC.Garble}(\mathbf{keys}, C)$:

$$\mathsf{GC.Eval}(\widehat{C}, \{\mathsf{keys}_i[x_i]\}_{i\in[n]}) = C(x).$$

Security: There exists a pair of PPT algorithm (S_1, S_2), such that any PPT adversary A wins the following game with at most negligible advantage:

1. A gives a circuit C and an input x to the challenger;
2. The challenger flips a bit b.

If $b = 0$:

– It chooses random garbling key $\mathbf{keys} \leftarrow \mathsf{GC.Gen}(1^\lambda)$;
– It sets $(\widetilde{C} \leftarrow \mathsf{GC.Garble}(\mathbf{keys}, C; \sigma), \widetilde{x}_i = \mathsf{keys}_i[x_i](i \in [n])$;
– It sends $\widetilde{C}, \widetilde{x}, \mathbf{keys}, \sigma$ to the adversary.

If $b = 1$:
– It sets $y = C(x)$;
– It runs the simulator $(\widetilde{C}, \widetilde{x}, \mathsf{st}) \leftarrow S_1(C, y)$
– It runs the simulator $(\mathbf{keys}, \sigma) \leftarrow S_2(\mathsf{st}, x)$
– It sends $\widetilde{C}, \widetilde{x}, \mathbf{keys}, \sigma$ to the adversary.

3. The adversary outputs a bit b'.

The adversary wins if $b = b'$.

We recall that (equivocal) garbled circuit schemes can be constructed from one-way functions.

Terminology of Input Labels. We note that, labels in boldface **keys** refer to all labels corresponding to all input wires. keys_i refers to the two input labels of the i-th wire and $\mathsf{keys}_i[b]$ refers to exactly one of them for $b \in \{0, 1\}$.

3.3 Equivocal Commitments

We define (adaptive) equivocal commitments (in the local CRS model).

Definition 3.2 (Non-interactive Equivocal Commitment). A *non-interactive equivocal commitment* scheme C is a tuple of five polynomial-time algorithms $\mathsf{C} = (\mathsf{C.Setup}, \mathsf{C.Setup_{equiv}}, \mathsf{C.Com}, \mathsf{C.Sim}, \mathsf{C.Equiv})$

Setup: $\mathrm{ck} \leftarrow \mathsf{C.Setup}(1^\lambda)$ expects as input the unary representation of the security parameter λ and outputs a public parameter ck.

Equivocal Setup: $(\mathrm{ck}, \mathsf{trap_q}) \leftarrow \mathsf{C.Setup_{equiv}}(1^\lambda)$ outputs a public parameter ck together with a trapdoor $\mathsf{trap_q}$ (used for equivocation).

Commitment: $\mathsf{com} = \mathsf{C.Com}(\mathrm{ck}, x; r)$ generates a commitment com of $x \in \{0,1\}^{\mathrm{poly}(\lambda)}$ using random tape $x \in \{0,1\}^{\mathrm{poly}(\lambda)}$;

Simulation: $(\mathsf{com}, \mathsf{st^c}) = \mathsf{C.Sim}(\mathrm{ck}, \mathsf{trap_q})$ outputs a simulated commitment and a state used to equivocate the commitment;

Equivocation: $\tilde{r} = \mathsf{C.Equiv}(\mathrm{ck}, \mathsf{trap_q}, \mathsf{com}, \mathsf{st^c}, x)$ equivocates the commitment com to open to x;

satisfying the following properties:

Equivocality: For any polynomial-time circuit family $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a negligible function negl, such that for any $\lambda \in N$:

$$\left| \Pr\left[A_\lambda(\mathsf{st}, \mathsf{com}, r) = 1 : \begin{array}{l} (\mathrm{ck}, \mathsf{trap_q}) \leftarrow \mathsf{C.Setup_{equiv}}(1^\lambda); \\ (x, \mathsf{st}) \leftarrow A(\mathrm{ck}); \\ \mathsf{com} \leftarrow \mathsf{C.Com}(1^\lambda, x; r) \end{array} \right] \right.$$
$$\left. - \Pr\left[A_\lambda(\mathsf{st}, \mathsf{com}, \tilde{r}) = 1 : \begin{array}{l} (\mathrm{ck}, \mathsf{trap_q}) \leftarrow \mathsf{C.Setup_{equiv}}(1^\lambda); \\ (x, \mathsf{st}) \leftarrow A(\mathrm{ck}); \\ (\mathsf{com}, \mathsf{st^c}) \leftarrow \mathsf{C.Sim}(\mathrm{ck}, \mathsf{trap_e}); \\ \tilde{r} \leftarrow \mathsf{C.Equiv}(\mathrm{ck}, \mathsf{trap_e}, \mathsf{com}, \mathsf{st^c}, x) \end{array} \right] \right| \leq \mathrm{negl}(\lambda).$$

Binding: For any polynomial-time circuit family $A = \{A_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a negligible function negl, such that for any $\lambda \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l}\mathsf{C.Com}(x_0; r_0) = \mathsf{C.Com}(x_1; r_1)\\ \text{and } x_0 \neq x_1\end{array} : \begin{array}{l}\mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda);\\ (x_0, r_0, x_1, r_1) \leftarrow A_\lambda(\mathsf{ck})\end{array}\right] \leq \mathrm{negl}(\lambda).$$

Indistinguishability of Public Parameters: We require that the two following distributions are computationally indistinguishable:

$$\{\mathsf{ck} : \mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda)\}, \qquad \{\mathsf{ck} : (\mathsf{ck}, \mathsf{trap_q}) \leftarrow \mathsf{C.Setup_{equiv}}(1^\lambda)\}.$$

Claim. Assuming the existence one-way functions, there exist equivocal commitments.

Proof. We can use the construction that is implicit in Appendix B of the full version of [28], using a pseudorandom generator G from $\{0,1\}^\lambda$ to $\{0,1\}^{3\lambda}$:

Setup: $\mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda)$ outputs a uniform string $\mathsf{ck} \in \{0,1\}^{3\lambda}$.
Equivocal Setup: $(\mathsf{ck}, \mathsf{trap_q}) \leftarrow \mathsf{C.Setup_{equiv}}(1^\lambda)$ generates a pair of uniform strings $\mathsf{trap_q} = (\mathsf{trap_{q_0}}, \mathsf{trap_{q_1}}) \in \{0,1\}^{2\lambda}$ and sets $\mathsf{ck} = G(\mathsf{trap_{q_0}}) \oplus G(\mathsf{trap_{q_1}})$.
Commitment: $\mathsf{com} = \mathsf{C.Com}(\mathsf{ck}, x; r)$ with $r \in \{0,1\}^\lambda$, sets $\mathsf{com} = G(r)$ if $x = 0$ and $\mathsf{com} = G(r) \oplus \mathsf{ck}$ if $x = 1$ (assuming messages x are bits, extension to strings is straightforward by parallel repetition).
Simulation: $(\mathsf{com}, \mathsf{st^c}) = \mathsf{C.Sim}(\mathsf{ck}, \mathsf{trap_q})$ sets $\mathsf{com} = G(\mathsf{trap_{q_0}})$ and $\mathsf{st^c} = \perp$.
Equivocation: $\tilde{r} = \mathsf{C.Equiv}(\mathsf{ck}, \mathsf{trap_q}, \mathsf{com}, \mathsf{st^c}, x)$ returns $\tilde{r} = \mathsf{trap_{q_0}}$ if $x = 0$ and $\tilde{r} = \mathsf{trap_{q_1}}$ if $x = 1$.

Binding comes from the fact that with overwhelming probability over $\mathsf{ck} \in \{0,1\}^{3\lambda}$, there does not exist r_0 and r_1 such that $G(r_0) \oplus G(r_1) = \mathsf{ck}$. Indistinguishability of public parameters and equivocality follows from the security of the pseudorandom generator G. ■

3.4 (Augmented) Non-committing Encryption

Let us now recall the definitions of non-committing encryption (NCE) and augmented NCE from [12,15].

Definition 3.3 (Non-committing encryption). A non-committing (bit) encryption scheme (NCE) consists of a tuple $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec}, \mathsf{NC.Sim})$ where $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec})$ is an encryption scheme and $\mathsf{NC.Sim}$ is the simulation satisfying the following property: for $b \in \{0,1\}$ the following distributions are computationally indistinguishable:

$$\{(\mathrm{pk}, c, \rho_G, \rho_E) : (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{NC.Gen}(1^\lambda; \rho_G), c = \mathsf{NC.Enc_{pk}}(b; \rho_E)\}_{\lambda,b},$$
$$\{(\mathrm{pk}, c, \rho_G^b, \rho_E^b) : (\mathrm{pk}, c, \rho_G^0, \rho_E^0, \rho_G^1, \rho_E^1) \leftarrow \mathsf{NC.Sim}(1^\lambda)\}_{\lambda,b}.$$

Definition 3.4 (Augmented non-committing encryption). An augmented non-committing encryption scheme (NCE) consists of a tuple $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec}, \mathsf{NC.Sim}, \mathsf{NC.Gen_{Obl}}, \mathsf{NC.Gen_{Inv}})$ where $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec}, \mathsf{NC.Sim})$ is an NCE and:

Oblivious Sampling: $\mathsf{NC.Gen}_{\mathsf{Obl}}(1^\lambda)$ obliviously generates a public key pk (without knowing the associated secret key sk.

Inverse Key Sampling: $\mathsf{NC.Gen}_{\mathsf{Inv}}(\mathrm{pk})$ explains the randomness for the key pk.

satisfying the following property:

Obliviousness: The following distributions are indistinguishable:

$$\{(\mathrm{pk}, \rho) : \mathrm{pk} \leftarrow \mathsf{NC.Gen}_{\mathsf{Obl}}(1^\lambda; \rho)\}_\lambda,$$
$$\{(\mathrm{pk}, \tilde{\rho}) : (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{NC.Gen}(1^\lambda);\ \tilde{\rho} \leftarrow \mathsf{NC.Gen}_{\mathsf{Inv}}(\mathrm{pk})\}_\lambda.$$

4 Definitions of UC Adaptive MPC

4.1 General Definition of Universal Composability

We refer the reader to the full version and to [10] for the general definitions for UC security.

General Functionality. We consider the general-UC N-party functionality $\mathcal{F}$, which securely evaluates any polynomial-time (possibly randomize) function $f : (\{0,1\}^{\ell_{in}})^N \rightarrow (\{0,1\}^{\ell_{out}})^N$. The functionality $\mathcal{F}_f$ is parameterized with a function f.

From Deterministic to Randomized Functionalities. Our multi-party Protocol 1 UC-securely realizes the general functionality $\mathcal{F}_f$ when the function f is restricted to be any deterministic poly-time function with N inputs and single output. This functionality is defined in Fig. 1. Standard techniques allow to obtain a protocol that UC-securely realizes the general functionality $\mathcal{F}_f$ for any function f. See details in the full version.

Adversarial Model. A *static* adversary $\mathcal{A}$ chooses the set of corrupted parties before the protocol starts, as opposed to an *adaptive* adversary that can corrupt the players during the protocol. We say that the adversary is *semi-honest* if $\mathcal{A}$ follows the protocol but tries to extract some information about the other parties' inputs from his view of the protocol. We say that the adversary is *malicious* if $\mathcal{A}$ is allowed to deviate arbitrarily from the protocol specifications. We will say that a protocol is *semi-honest-secure* if it is secure against a semi-honest adversary and *malicious-secure* if it is secure against a malicious adversary. In this work, we consider malicious security against an adaptive adversary.

Communication Channel. In our results we consider a secure simultaneous message exchange channel in which all parties can simultaneously send messages over the channel at the same communication round in the presence of a *rushing* adversary. In every communication round, a rushing adversary sees the messages from the honest parties and only then chooses the messages on behalf of the malicious parties. For simplicity, we assume that the parties can broadcast messages and have authenticated channels. This can be achieved using standard methods.

Functionality $\mathcal{F}_f$

$\mathcal{F}_f$ parameterized by an N-ary deterministic single output function f, running with parties $\mathcal{P} = \{P_1, \ldots P_N\}$ (of which some may be corrupted) and an adversary $\mathcal{S}$, proceeds as follows:

1. Each party P_i (and $\mathcal{S}$ on behalf of P_i if P_i is corrupted) sends $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$ to the functionality. Upon receiving $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$, record the tuple (P_i, x_i), send the message $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i)$ to the adversary $\mathcal{S}$, and ignore subsequent such messages for the same P_i.
2. Upon receiving the inputs from all parties, evaluate $y \leftarrow f(x_1, \ldots, x_N)$. Send the message $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, y)$ to the adversary $\mathcal{S}$ if at least one party is corrupted, and the message $(\mathsf{output}, \mathsf{sid}, \mathcal{P})$ if no party is corrupted.
3. On receiving $(\mathsf{deliver}, \mathsf{sid}, \mathcal{P}, P_i)$ from $\mathcal{S}$, output $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, y)$ to P_i. (And ignore the message if inputs from all parties in $\mathcal{P}$ have not been received or if such a message has already been received for this party P_i.)

Fig. 1. General functionality for deterministic single output functionalities.

4.2 The Local CRS Model

In the common reference string (CRS) model [13,15], all parties in the system obtain from a trusted party a reference string, which is sampled according to a pre-specified distribution D. The reference string is referred to as the *CRS*. In the UC framework, this is modeled by an ideal functionality $\mathcal{F}_{CRS}^{D}$ that samples a string ρ from a pre-specified distribution D and sets ρ as the CRS. $\mathcal{F}_{CRS}^{D}$ is described in Fig. 2.

Functionality $\mathcal{F}_{\mathbf{CRS}}^{\mathbf{D}}$

1. Upon activation with session id *sid* proceed as follows. Sample $\rho = D(r)$, where r denotes uniform random coins, and send $(\mathtt{crs}, sid, \rho)$ to the adversary.
2. On receiving $(\mathtt{crs}, sid)$ from some party send $(\mathtt{crs}, sid, \rho)$ to that party.

Fig. 2. The common reference string functionality.

5 Two-Round UC Adaptive-OT

5.1 Definition of Oblivious Transfer

(Two-out-of-one) oblivious transfer is a two-party functionality, involving a sender S with input x_0, x_1, and a receiver R with input $\sigma \in \{0,1\}$. R learns x_σ (or $\perp$ if the protocol fails) and nothing else. S learns nothing about σ. The definition of the ideal oblivious transfer functionality, denoted by $\mathcal{F}_{\mathsf{OT}}$, appears in Fig. 3.

Functionality $\mathcal{F}_{\mathsf{OT}}$

$\mathcal{F}_{\mathsf{OT}}$ running with an oblivious transfer sender S, a receiver R and an adversary $\mathcal{S}$ proceeds as follows:

1. Upon receiving a message $(\mathsf{sender}, \mathsf{sid}, x_0, x_1)$ from S, where each $x_i \in \{0,1\}^\lambda$, record the tuple (x_0, x_1), send the message $(\mathsf{sender}, \mathsf{sid})$ to the adversary $\mathcal{S}$, and ignore subsequent such messages.
2. Upon receiving a message $(\mathsf{receiver}, \mathsf{sid}, \mathsf{R}, \sigma)$ from R, where $\sigma \in \{0,1\}$, check if a $(\mathsf{sender}, \mathsf{sid}, x_0, x_1)$ message was previously received. If no such message was received, do nothing. Otherwise, send $(\mathsf{output}, \mathsf{sid})$ to the adversary $\mathcal{S}$ if the receiver R is not corrupted, and send $(\mathsf{output}, \mathsf{sid}, x_\sigma)$ to the adversary $\mathcal{S}$.
3. Upon receiving $(\mathsf{deliver}, \mathsf{sid}, \mathsf{R})$ from $\mathcal{S}$, output (sid, x_σ) to R. (And ignore the message if inputs from all parties have not been received or if such a message has already been received.)
4. Upon receiving $(\mathsf{deliver}, \mathsf{sid}, \mathsf{S})$ from $\mathcal{S}$, output (sid) to S. (And ignore the message if inputs from all parties have not been received or if such a message has already been received.)

Fig. 3. Oblivious transfer functionality.

Adversarial Model. Our construction of 2-round OT secure against adaptive corruption will start with 2-round OT that is only secure against static corruption and has certain special properties, and gradually transform this property to handle different adaptive corruption scenario. We list all the corruption scenarios we consider below.

1. *Static corruption* where the adversary chooses the corrupted parties at the beginning of the protocol execution.
2. *Sender-semi-adaptive corruption* where the adversary statically corrupts the receiver from the beginning and adaptively chooses whether and when to corrupt the *sender* during the execution of the protocol.
3. *Receiver-semi-adaptive corruption* where the adversary statically corrupts the sender from the beginning and adaptively chooses whether and when to corrupt the *receiver* during the execution of the protocol.
4. *Semi-adaptive corruption* where the adversary either performs sender-semi-adaptive corruption or receiver-semi-adaptive corruption. In other words, the adversary always corrupts one party from the beginning and adaptively chooses whether and when to corrupt the other party during the execution.
5. *Adaptive corruption* where the adversary adaptively chooses whether and when to corrupt any party during the execution. Note that adaptive corruption covers semi-adaptive corruption, as well as the scenarios where the receiver and/or sender are corrupted after the entire execution is complete.

Two-Round Oblivious Transfer Protocols. In this work, we consider 2-round oblivious transfer protocols, denoted as $\Pi = \langle \mathsf{S}, \mathsf{R} \rangle$. For convenience, we often use S and R to refer to the sender and the receiver. We also use them to denote

the sender and receiver algorithms, where the sender's algorithm $\mathsf{S}(\mathsf{sid}, x_0, x_1)$ takes input a session id and two input strings, and receiver's algorithm $\mathsf{R}(\mathsf{sid}, \sigma)$ takes input a session id and a selection bit. Below, for convenience of notation, in context where the session id is clear, or can be arbitrary, we suppress sid from the algorithms. For the cases where we consider 2-round oblivious transfer in the CRS-hybrid model, we denote by K the CRS algorithm generation. We denote by $\mathcal{S}_{\mathsf{R}}$ ($\mathcal{S}_{\mathsf{S}}$) the ideal world simulator for $\mathcal{F}_{\mathsf{OT}}$ simulating the view of an adversarial receiver (sender).

5.2 Oblivious Sampling

Definition 5.1 (Receiver-oblivious-sampleability). A 2-round oblivious transfer protocol with receiver oblivious sampleability is a 2-round oblivious OT protocol $(\Pi = \langle \mathsf{S}, \mathsf{R} \rangle, \mathsf{K})$ with the additional algorithms $(R_{\mathsf{Obl}}, R_{\mathsf{Inv}})$, such that for any bit $\sigma \in \{0,1\}$, the following two distributions are computationally indistinguishable:

$$\{(\mathrm{crs}, \tilde{\mu}, \tilde{\rho}) \ : \ \mathrm{crs} \leftarrow \mathsf{K}(1^\lambda);\ \tilde{\rho} \leftarrow \{0,1\}^\lambda;\ \tilde{\mu} \leftarrow \mathsf{R}_{\mathsf{Obl}}(\mathrm{crs}; \tilde{\rho})\},$$
$$\{(\mathrm{crs}, \mu, \rho) \ : \ \mathrm{crs} \leftarrow \mathsf{K}(1^\lambda);\ \mu = \mathsf{R}(\mathrm{crs}, \sigma); \rho \leftarrow \mathsf{R}_{\mathsf{Inv}}(\mathrm{crs}, \mu)\}.$$

Definition 5.2 [Sender-oblivious-sampleability]**.** A 2-round oblivious transfer protocol with sender oblivious sampleability is a 2-round oblivious OT protocol $(\Pi = \langle \mathsf{S}, \mathsf{R} \rangle, \mathsf{K})$ with the additional algorithms $(\mathsf{S}_{\mathsf{Obl}}, \mathsf{S}_{\mathsf{Inv}})$ such that, for any message $x_0, x_1 \in \{0,1\}^\lambda$, no PPT adversary $\mathcal{A}$ (acting as a malicious receiver), can distinguish the following two experiments:

Real-world experiment:

1. A challenger $\mathcal{C}$ runs the simulator $\mathcal{S}_{\mathsf{R}}$ of Π, which interacts with $\mathcal{A}$ in a straight-line: (i) $\mathcal{S}_{\mathsf{R}}$ simulates the CRS crs for $\mathcal{A}$; (ii) when $\mathcal{A}$ sends a first OT message μ, $\mathcal{S}_{\mathsf{R}}$ extracts from μ a selection bit σ.
2. $\mathcal{C}$ runs S to obtain an obliviously sampled OT second message $\nu \leftarrow \mathsf{S}_{\mathsf{Obl}}(\mathrm{crs}, \mu; \tilde{\rho})$, picks a random string $t \leftarrow \{0,1\}^\lambda$, and sends to $\mathcal{A}$ the selection bit σ, the message ν, and the string t.
3. $\mathcal{C}$ sends $\tilde{\rho}$ to $\mathcal{A}$.

Simulated-world experiment:

1. A challenger $\mathcal{C}$ runs the simulator $\mathcal{S}_{\mathsf{R}}$ of Π, which interacts with $\mathcal{A}$ in a straight-line: (i) $\mathcal{S}_{\mathsf{R}}$ simulates the CRS crs for $\mathcal{A}$; (ii) when $\mathcal{A}$ sends a first OT message μ, $\mathcal{S}_{\mathsf{R}}$ extracts from μ a selection bit σ.
2. $\mathcal{C}$ runs S to obtain an honestly generated OT second message $\nu \leftarrow \mathsf{S}(\mathrm{crs}, \mu, t_0, t_1)$ for $t_0, t_1 \leftarrow \{0,1\}^\lambda$ and sends to $\mathcal{A}$ the selection bit σ, the message ν, and the string $t_{1-\sigma}$.
3. $\mathcal{C}$ computes $\rho \leftarrow \mathsf{S}_{\mathsf{Inv}}(\mathrm{crs}, \nu)$ and sends ρ to $\mathcal{A}$.

5.3 Transformation 1: Achieving Sender Equivocality

Proposition 5.3. *Assume the existence of two-round oblivious transfer with the following properties:*

- *UC-Security against static receiver corruption by a malicious adversary.*
- *UC-Security against static sender corruption by a semi-honest adversary.*
- *Sender oblivious sampleability.*

Then, there exists a two-round oblivious transfer protocol in the CRS-hybrid model with the following properties:

- *UC-Security against static receiver corruption and post-execution sender corruption (or UC-Security against sender-semi-adaptive corruption for short) by a malicious adversary.*

Additionally, the compilation preserves (1) receiver-oblivious-sampleability and (2) UC-Security against static sender corruption by a semi-honest adversary, if the original protocol satisfies either of the properties.

Our Protocol. In this section we will present our UC oblivious transfer protocol Π_{OT} secure against sender-semi-adaptive corruption, described in Fig. 4. For simplicity of exposition, in the sequel, we will assume that random coins are an implicit input to the sender and receiver algorithms, unless specified explicitly. The security proof is provided in the full version.

Protocol Π_{OT}

Let $(\Pi = \langle \mathsf{S}, \mathsf{R} \rangle, \mathsf{K}, \mathsf{S}_{\mathsf{Obl}})$ be an oblivious transfer protocol with an oblivious sender algorithm $\mathsf{S}_{\mathsf{Obl}}$.

Common Reference String: Generate $\mathsf{crs} \leftarrow \mathsf{K}(1^\lambda)$.
Inputs: Sender holds two strings $x_0, x_1 \in \{0,1\}$ and receiver holds a bit σ.

1. Given input $(\mathsf{receiver}, \mathsf{sid}, \sigma)$, receiver R_{OT} runs S on input $(\mathsf{receiver}, \mathsf{sid}, \sigma)$ to obtain the message (sid, μ) which R_{OT} sends to S_{OT}.
2. Given input $(\mathsf{sender}, \mathsf{sid}, x_0, x_1)$ and message (sid, μ), sender S_{OT} generates random strings $r_0, r_1, s_0, s_1 \in \{0,1\}^\lambda$ and generates the following:

$$\nu_{0,x_0} = \mathsf{S}(\mathrm{crs}, \mu, r_0, s_0) \qquad \nu_{0,1-x_0} = \mathsf{S}_{\mathsf{Obl}}(\mathrm{crs}, \mu)$$
$$\nu_{1,x_1} = \mathsf{S}(\mathrm{crs}, \mu, s_1, r_1) \qquad \nu_{1,1-x_1} = \mathsf{S}_{\mathsf{Obl}}(\mathrm{crs}, \mu)$$

   and sends $(\mathsf{sid}, r_0, r_1, \nu_{0,0}, \nu_{0,1}, \nu_{1,0}, \nu_{1,1})$ to R_{OT}.
3. Upon receiving the message $(\mathsf{sid}, r_0, r_1, \nu_{0,0}, \nu_{0,1}, \nu_{1,0}, \nu_{1,1})$, R_{OT} feeds R with $(\mathsf{sid}, \nu_{\sigma,0})$ and $(\mathsf{sid}, \nu_{\sigma,1})$ in two parallel invocations to obtain y and y'. If $y = r_\sigma$, it outputs $(\mathsf{sid}, 0)$, and if $y' = r_\sigma$ it outputs $(\mathsf{sid}, 1)$.

Fig. 4. Sender-semi-adaptive oblivious transfer $\Pi_{\mathsf{OT}} = \langle \mathsf{S}_{\mathsf{OT}}, \mathsf{R}_{\mathsf{OT}} \rangle$ protocol.

5.4 Transformation 2: Achieving Receiver Equivocality Against Malicious Sender

Proposition 5.4. *Assume the existence of two-round oblivious transfer with the following properties:*

- *UC-Security against static sender corruption by a semi-honest adversary.*
- *UC-Security against a static receiver corruption and post-execution sender corruption (or UC-Security against sender-semi-adaptive corruption for short) by a malicious adversary.*
- *Receiver-Oblivious-sampleability.*

Then there exists a two-round oblivious transfer protocol in the CRS-hybrid model with the following properties:

- *UC-Security against semi-adaptive corruption by a malicious adversary.*

Our Protocol. In this section we will present our UC oblivious transfer protocol Π_{OT} secure against semi-adaptive corruption, described in Fig. 5. The security proof is provided in the full version.

5.5 Transformation 3: From Semi-Adaptive-OT to Adaptive-OT

Proposition 5.5. *Assume the existence of augmented non-committing encryption and two-round oblivious transfer with the following property:*

- *UC-Security against semi-adaptive corruption by a malicious adversary.*

Then there exists a two-round oblivious transfer protocol with the following property:

- *UC-Security against adaptive corruption.*

Our Protocol. In this section we will present our UC oblivious transfer protocol Π_{OT} secure against adaptive corruptions, described in Fig. 6. For simplicity of exposition, in the sequel, we will assume that random coins are an implicit input to the sender and receiver algorithms, unless specified explicitly. Furthermore, to simplify notation, we suppose that the sender's inputs are bits. Extension to strings is straightforward: it just requires to use string NCE instead of bit NCE (which can be constructed by parallel repetition of bit NCE). The security proof is provided in the full version.

5.6 Instantiation of Static-OT with Oblivious Sampleability

In the full version, we show instantiations of static-OT with oblivious sampleability from the DDH and the QR assumptions. We also construct a slightly weaker variant of 2-round static-OT (called *half-OT*) with oblivious sampleability from LWE using a variant of the previous generic construction. Finally, we provide a variant of Transformation 1 (Sect. 5.3) that starts from a half-OT instead of a static-OT.

Protocol Π_{OT}

Let $(\Pi = \langle \mathsf{S}, \mathsf{R}\rangle, \mathsf{K}, \mathsf{R}_{\mathsf{Obl}})$ be a UC static receiver corruption and sender-semi-adaptive corruption oblivious transfer protocol with an oblivious receiver algorithm $\mathsf{R}_{\mathsf{Obl}}$, let $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec}, \mathsf{NC.Sim})$ be a somewhat NCE scheme, let $\mathsf{GC} = (\mathsf{GC.Gen}, \mathsf{GC.Garble}, \mathsf{GC.Eval})$ be an equivocal garbling scheme and let $\mathsf{C} = (\mathsf{C.Setup}, \mathsf{C.Com})$ be a non-interactive equivocal commitment scheme.

Common Reference String: Generate $\mathsf{crs}' \leftarrow \mathsf{K}(1^\lambda)$, $\mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda)$ and set $\mathsf{crs} = (\mathsf{crs}', \mathsf{ck})$.
Inputs: Sender holds two strings $x_0, x_1 \in \{0,1\}$ and receiver holds a bit σ.

1. Given input $(\mathsf{receiver}, \mathsf{sid}, \sigma)$, receiver R_{OT} generates commitment $\mathsf{com} = \mathsf{C.Com}\big(\sigma; (r_1 \| r_2 \| ... \| r_T)\big)$ where $r_i \in \{0,1\}$ and an NCE key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{NC.Gen}(1^\lambda)$. Send pk and for all $i \in [\kappa]$, R_{OT} invokes R to generate and send to S_{OT} the following:

$$\mu_{\delta,i} = \begin{cases} \mathsf{R}(\mathsf{crs}, r_i) & \text{if } \delta = \sigma \\ \mathsf{R}_{\mathsf{Obl}}(\mathsf{crs}) & \text{otherwise} \end{cases}$$

2. Given input $(\mathsf{sender}, \mathsf{sid}, x_0, x_1)$ and messages $(\mathsf{sid}, \{\mu_{\delta,i}\})$, sender S_{OT} proceeds as follows:

   (a) For $\delta \in \{0,1\}$ generate circuit C_δ as follows:

   $$C_\delta(r_1 \| r_2 \| ... \| r_T) = \Big\{ \text{output } x_\delta \;\; \text{if } \mathsf{com} = \mathsf{C.Com}\big(\delta; (r_1 \| r_2 \| ... \| r_T)\big) \Big\}$$

   (b) Generate the garble circuit $\widehat{\mathrm{C}}_\delta$, with input labels $\mathbf{keys}^\delta \leftarrow \mathsf{GC.Gen}(1^\lambda)$:

   $$\widehat{\mathrm{C}}_\delta \leftarrow \mathsf{GC.Garble}(\mathbf{keys}^\delta, C_\delta)$$

   (c) For $\delta \in \{0,1\}$ and $i \in [T]$ generate $\nu_{\delta,i} = \mathsf{S}(\mathsf{crs}, \mu_{\delta,i}, \mathsf{keys}_i^\delta)$;

   and sends $\big(\mathsf{sid}, \mathsf{ct} \leftarrow \mathsf{NC.Enc}(\mathsf{pk}, \{\widehat{\mathrm{C}}_\delta, \nu_{\delta,i}\}_{\delta\in\{0,1\}, i\in[T]})\big)$ to R_{OT}.
3. Upon receiving the message $(\mathsf{sid}, \mathsf{ct})$ compute $\{\widehat{\mathrm{C}}_\delta, \nu_{\delta,i}\}_{\delta\in\{0,1\}, i\in[T]} = \mathsf{NC.Dec}(\mathsf{sk}, \mathsf{ct})$, R_{OT} feeds R with $(\mathsf{sid}, \nu_{\delta,i})$ for all $i \in [T]$ to obtain the labels $\{\mathsf{keys}_i^\delta[r_i]\}$ corresponding to r_i. Next, it evaluates the garbled circuits to get $x_\delta = \mathsf{GC.Eval}(\widehat{C}_\delta, \mathsf{keys}_i^\delta[r_i])$. If $x_\delta \neq \perp$ for at least one $\delta \in \{0,1\}$ then output (sid, x_δ) else output $\perp$.

Fig. 5. Semi-adaptive oblivious transfer $\Pi_{\mathsf{OT}} = \langle \mathsf{S}_{\mathsf{OT}}, \mathsf{R}_{\mathsf{OT}}\rangle$ protocol.

6 Two-Round UC Adaptive-MPC

In this section we upgrade the static construction of [5] to the adaptive setting. The changes we make to the construction of [5] is to lift the security of the garble circuit and oblivious transfer schemes to the adaptive setting. Unlike [5], we also obtain security against adaptive malicious adversaries without NIZK.

Protocol Π_{OT}

Let $(\Pi = \langle \mathsf{S}, \mathsf{R} \rangle, \mathsf{K})$ be a UC-secure semi-adaptive oblivious transfer protocol. Let $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec}, \mathsf{NC.Sim}, \mathsf{NC.Gen}_{\mathsf{Obl}}, \mathsf{NC.Gen}_{\mathsf{Inv}})$ be an augmented NCE scheme.

Common Reference String: Generate $\mathsf{crs} \leftarrow \mathsf{K}(1^\lambda)$.
Inputs: Sender holds two bits $x_0, x_1 \in \{0,1\}$ and receiver holds a bit σ.

1. Given input $(\mathsf{receiver}, \mathsf{sid}, \sigma)$, receiver R_{OT} does the following:
   (a) run R on input $(\mathsf{receiver}, \mathsf{sid}, \sigma)$ to obtain the message (sid, μ).
   (b) generate an NCE key pair $(\mathrm{pk}_\sigma, \mathrm{sk}_\sigma) \leftarrow \mathsf{NC.Gen}(1^\lambda)$.
   (c) obliviously sample an NCE public key $\mathrm{pk}_{1-\sigma} \leftarrow \mathsf{NC.Gen}_{\mathsf{Obl}}(1^\lambda)$.
   (d) send $(\mathsf{sid}, \mu, \mathrm{pk}_0, \mathrm{pk}_1)$ to the sender S_{OT}.
2. Given input $(\mathsf{sender}, \mathsf{sid}, x_0, x_1)$ and message (sid, μ), sender S_{OT} does the following:
   (a) pick two random strings $r_0, r_1 \leftarrow \{0,1\}^\lambda$
   (b) run S on input $(\mathsf{sender}, \mathsf{sid}, r_0 \oplus x_0, r_1 \oplus x_1)$ to obtain the message (sid, ν).
   (c) encrypts x_0 and x_1 under pk_0 and pk_1 respectively: $\mathrm{ct}_0 \leftarrow \mathsf{NC.Enc}(\mathrm{pk}_0, x_0)$ and $\mathrm{ct}_1 \leftarrow \mathsf{NC.Enc}(\mathrm{pk}_1, x_1)$.
   (d) send $(\mathsf{sid}, \nu, \mathrm{ct}_0, \mathrm{ct}_1)$ to the receiver R_{OT}.
3. Upon receiving the message $(\mathsf{sid}, \nu, \mathrm{ct}_0, \mathrm{ct}_1)$, R_{OT} feeds R with (sid, ν) to obtain y. It also decrypts ct_σ into r: $r = \mathsf{NC.Dec}(\mathrm{sk}_\sigma, \mathrm{ct}_\sigma)$ and output $y \oplus r$.

Fig. 6. Adaptive oblivious transfer $\Pi_{\mathsf{OT}} = \langle \mathsf{S}_{\mathsf{OT}}, \mathsf{R}_{\mathsf{OT}} \rangle$ protocol.

Protocol Π_{MPC}. We provide an intuitive description of the protocol. A formal description appears in Protocol 1. The main idea is to collapse a constant L-round adaptive N-party protocol π secure against malicious adversaries into a two-round protocol based on equivocal garbled circuits and adaptive oblivious transfer. The first round of the protocol acts as a catalyst for a virtual execution of π via equivocal garbled circuits sent by all the parties in the second round. In particular, each party P_i garbles their next-step circuit $\mathsf{Nextmsg}_i(x_i, r_i, \star)$ in an execution of the inner protocol π computing the desired functionality f. The next-step circuit contains hardcoded the input and randomness (x_i, r_i) of party P_i and produces P_i's next message m_i^ℓ for round ℓ on input the messages received from all parties in all previous rounds $M^{<\ell} = \{m_j^{\ell'}\}_{j \in [N], \ell' < \ell}$. We denote these circuits by $\widehat{\mathrm{F}}_i^\ell$. These garbled circuits expect as input messages from other parties as well as output messages for other garbled circuits. There are certain barrier to put this idea into practice. First of all, parties can perform residual attacks on the honest parties inputs. To overcome this barrier, we use the first round to "bind" the parties to their inputs via an oblivious transfer protocol. Next, each party in the second round needs to generate verification circuits $\widehat{\mathrm{V}}_{i,j}$ that take as input a proof for each other party's input message and verify that the message is honestly generated from the inputs and random tapes committed in

the first round. This ensures that only the unique sequence of honestly generated messages is accepted by honest parties' $\widehat{\mathrm{F}}_i^\ell$ garbled circuits. Our protocol below describes how to combine the above ideas.

6.1 The Protocol

In this section we present our adaptively secure two-round MPC protocol secure against malicious adversaries, described in Protocol 1.

Protocol 1 (Adaptive malicious protocol Π_{MPC}). Let f be an arbitrary N-party functionality. Protocol Π_{MPC} relies on the following components:

- An adaptive malicious constant L-round N-party protocol $\pi = (\mathsf{Setup}_\pi, \mathsf{Nextmsg}, \mathsf{Output})$ for f. Setup_π generates the CRS crs_π which is an implicit input of $\mathsf{Nextmsg}$ and Output. Without loss of generality, we will assume that in each round ℓ of π, each party P_i broadcasts a single message that depends on its input x_i, randomness r_i and on the messages $M^{<\ell} = \{m_j^{\ell'}\}_{j\in[N],\ell'<\ell}$ that it received from all parties in all previous rounds such that $m_j^\ell = \mathsf{Nextmsg}_j(x_j, r_j, M^{<\ell})$. $\mathsf{Nextmsg}_j$ is the next message function that computes the message broadcast by P_j. In the last round L of π each party P_i locally computes the output $y_i = \mathsf{Output}_i(x_i, r_i, M)$ after receiving all the messages $M = \{m_j^\ell\}_{j\in[N],\ell\in[L]}$.
- A malicious adaptive OT protocol $(\Pi = \langle S, R\rangle, \mathsf{K})$ where K is the OT setup algorithm.
- An equivocal garbling scheme $\mathsf{GC} = (\mathsf{GC.Gen}, \mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$.

Common Reference String: Generate $\mathrm{crs}_{\mathsf{OT}} \leftarrow \mathsf{K}(1^\lambda)$ and $\mathrm{crs}_\pi \leftarrow \mathsf{Setup}_\pi(1^\lambda, 1^N)$.[6] Set the CRS to be $\mathrm{crs} = (\mathrm{crs}_{\mathsf{OT}}, \mathrm{crs}_\pi)$.
Input: Parties $P_1, \ldots, P_N$ are given input $(x_1, \ldots, x_N)$, respectively.

- Round 1: For ℓ from L to 1 each party $P_{i^\star}$ proceeds as follows:
  1. Generate input labels $\mathbf{cKeys}_{i^\star}^\ell \leftarrow \mathsf{GC.Gen}(1^\lambda)$.
  2. Garble a *commitment* circuit $C_{i^\star}^\ell = U_\lambda(\star, (x_{i^\star}, r_{i^\star}))$, which is the universal circuit (with input size T) partially evaluated on $(x_{i^\star}, r_{i^\star})$: $\widehat{C}_{i^\star}^\ell \leftarrow \mathsf{GC.Garble}(\mathbf{cKeys}_{i^\star}^\ell, C_{i^\star}^\ell)$ where $r_{i^\star}$ is the random tape for running protocol π.
  3. For each $k \in [|\widehat{C}_{i^\star}^\ell|]$, generate OT receiver messages for the k-th bit of $\widehat{C}_{i^\star}^\ell$, denoted $|\widehat{C}_{i^\star}^\ell|_k$:
  $$\overline{\mu}_{i^\star,k}^\ell = \mathsf{R}(\mathrm{crs}_{\mathsf{OT}}, |C_{i^\star}^\ell|_k; \overline{\rho}_{i^\star,k}^\ell)$$
  4. For each $t \in [T]$, for each bit $b \in \{0,1\}$, generate OT receiver messages
  $$\mu_{i^\star,t,b}^\ell = \mathsf{R}(\mathrm{crs}_{\mathsf{OT}}, \mathbf{cKeys}_{i^\star,t}^\ell[b]; \rho_{i^\star,t,b}^\ell)$$

[6] Formally, we need a CRS $\mathrm{crs}_{\mathsf{OT}}$ for each instantiation of the OT protocol. For the sake of simplicity, we assume that there is a single CRS.

Output $c^{\ell}_{i^\star} = (\{\overline{\mu}^{\ell}_{i^\star,k}\}, \{\mu^{\ell}_{i^\star,t,b}\})$

– Round 2: For ℓ from L to 1 each party $P_{i^\star}$ garbles the *evaluation* circuits $\mathrm{F}_{i^\star} = \{\mathrm{F}^{\ell}_{i^\star}\}_{\ell\in[L]}$, defined in Fig. 7, as follows:

1. Generate input labels

$$\{\mathsf{cirKeys}^{\ell}_{i^\star,j}\}_{j\in[N]}, \mathsf{stateKeys}^{\ell}_{i^\star}, \{\mathsf{dataKeys}^{\ell}_{i^\star,j}\}_{j\in[N]} \leftarrow \mathsf{GC.Gen}(1^{\lambda}).$$

2. Garble the *evaluation* circuit $\mathrm{F}^{\ell}_{i^\star}$ and broadcast $\widehat{\mathrm{F}}_{i^\star} = \{\widehat{\mathrm{F}}^{\ell}_{i^\star}\}_{\ell\in[L]}$:

$$\widehat{\mathrm{F}}^{\ell}_{i^\star} \leftarrow \mathsf{GC.Garble}(\{\mathsf{cirKeys}^{\ell}_{i^\star,j}\}_j, \mathsf{stateKeys}^{\ell}_{i^\star}, \{\mathsf{dataKeys}^{\ell}_{i^\star,j}\}_j, \mathrm{F}^{\ell}_{i^\star}).$$

3. Generate OT sender messages on the received messages $\{\overline{\mu}^{\ell}_{j,k}\}_{j,k}$:

$$\overline{\nu}^{\ell}_{i^\star,j,k} = \mathsf{S}(\mathrm{crs}_{\mathsf{OT}}, \overline{\mu}^{\ell}_{j,k}, \mathsf{cirKeys}^{\ell}_{i^\star,j,k}[0], \mathsf{cirKeys}^{\ell}_{i^\star,j,k}[1]).$$

4. For each $k \in [|\widehat{C}^{\ell}_{i^\star}|]$ output the randomness $\overline{\rho}^{\ell}_{i^\star,k}$ used to generate $\overline{\mu}^{\ell}_{i^\star,k}$.

– Output phase: Each party evaluates the *evaluation* garbled circuits. In particular $P_{i^\star}$ proceeds as follows in L iterations ($\ell \in [L]$):

1. For all $i \in [N]$, $j \in [N]$, and $k \in [|\widehat{C}^{\ell}_{j}|]$, given $\overline{\rho}^{\ell}_{j,k}$ recover the labels $\mathsf{cirKeys}^{\ell}_{i,j,k}[b]$ corresponding to the bit $b = |\widehat{C}^{\ell}_{j}|_k$. For all $i \in [N]$, denote all the $[|\widehat{C}^{\ell}_{j}|]$ garble labels $\mathsf{cirKeys}^{\ell}_{i,j,k}[b]$ by $\gamma^{\ell}_{i,j}$.
2. If $\ell = 1$, for $i \in [N]$, evaluate the *evaluation* garble circuit $\mathsf{GC.Eval}(\widehat{\mathrm{F}}^{1}_{i}, \{\gamma^{1}_{i,j}\}_j)$ to obtain $(\mathsf{stateKeys}^{2}_{i}, \{\widehat{\mathrm{V}}^{1}_{i,j}, \nu^{1}_{i,j,t}, d^{1}_{i,t}\}_{j,t}, m^{1}_{i})$ for all $j \in [N]$ and $t \in [T]$. Note that for $\ell = 1$, $\mathsf{stateKeys}^{1}_{i}$ and $\{\mathsf{dataKeys}^{1}_{i,j}\}_j$ are the empty set.
3. For every $1 < \ell \le L$, for $i \in [N]$ and for each $j \in [N]$ proceed as follows. For all $t \in [T]$ set $g^{\ell-1}_{j,t} = |G^{\ell-1}_{j}|_t$ as the t-th bit of the circuit $G^{\ell-1}_{j}$. For simplicity of exposition, denote by $\alpha^{\ell-1}_{j,t} = \mathsf{cKeys}^{\ell-1}_{j,t}[b]$ the garble label of the *commitment* circuit corresponding to the bit $b = g^{\ell-1}_{j,t}$ and proceed as follows:
   (a) Given the randomness $d^{\ell-1}_{j,t}$, used to generate the Π_{OT} message $\nu^{\ell-1}_{i,j,t}$, recover the κ garble labels $\{\mathsf{vKeys}^{\ell-1}_{i,j,t'}[\alpha^{\ell-1}_{j,t}]\}_{t'}$ of the *verification* circuit where $(t-1)\cdot\kappa < t' \le t\cdot\kappa$. Denote all of the $\kappa \cdot T$ labels by $\beta^{\ell}_{i,j}$.
   (b) Evaluate the *verification* circuit $\mathsf{GC.Eval}(\widehat{\mathrm{V}}^{\ell-1}_{i,j}, \beta^{\ell}_{i,j})$ to receiver the garble labels corresponding to the message $m^{\ell-1}_{j}$ of the *evaluation* circuit i.e. $\mathsf{dataKeys}^{\ell}_{i,j}[m^{\ell-1}_{j}]$.
   (c) Evaluate the *evaluation* circuit $\mathsf{GC.Eval}(\widehat{\mathrm{F}}^{\ell}_{i}, \{\gamma^{\ell}_{i,j}\}_j, \mathsf{stateKeys}^{\ell}_{i}[M^{<\ell-1}], \{\mathsf{dataKeys}^{\ell}_{i,j}[m^{\ell-1}_{j}]\}_j)$ to obtain the values $(\mathsf{stateKeys}^{\ell+1}_{i}[M^{<\ell}], \{\widehat{\mathrm{V}}^{\ell}_{i,j}, \nu^{\ell}_{i,j,k}, d^{\ell}_{i,t}\}_{j,t}, m^{\ell}_{j})$ for the next round $\ell + 1$.

(d) For the case where $\ell = L$, the evaluation circuit outputs the empty set for the values $\mathsf{stateKeys}_i^{\ell+1}[M^{<\ell}]$ and $\{\widehat{\mathrm{V}}_{i,j}^{\ell}, \nu_{i,j,t}^{\ell}\}_j$.

4. After all L iterations, $P_{i^\star}$ obtains the set of all messages M, and computes the output $y_{i^\star} = \mathsf{Output}_{i^\star}(x_{i^\star}, r_{i^\star}, M)$.

Circuit F_i^{ℓ}

Hardwired Values: 1^λ, crs, ℓ, i, x_i, r_i, $\{\mu_{j,t,b}^{\ell}\}$, $\mathbf{stateKeys}_i^{\ell+1}$, $\{\mathbf{dataKeys}_{i,j}^{\ell+1}\}_{j\in[N]}$, $\{\rho_{i,t,b}^{\ell}\}_{t\in[T],b\in\{0,1\}}$.

Inputs: $(\{\widehat{C}_j^{\ell}\}_j, M^{<\ell-1}, \bar{m}^{\ell-1})$ where for $\ell > 1$:

- Garble labels corresponding to the circuits $\widehat{C}_j^{\ell}$ are denoted by $\mathbf{cirKeys}_{i,j}^{\ell}$.
- The input messages $M^{<\ell-1}$ are the messages of protocol π of the first $\ell-2$ rounds. Garble labels corresponding to this input are denoted by $\mathbf{stateKeys}_i^{\ell}$.
- The input messages $\bar{m}^{\ell-1} := \{m_j^{\ell-1}\}_{j\in[N]}$ are the $\ell-1$ round messages of protocol π. Garble labels corresponding to this input are denoted by $\mathbf{dataKeys}_{i,j}^{\ell}$.

Procedure:

1. Define the circuit G_j^{ℓ} as $G_j^{\ell}(\star,\star) = \mathsf{Nextmsg}_j(\star,\star,M^{<\ell-1},\bar{m}^{\ell-1})$, for $j \in [N]$ and set $g_{j,t} = |G_j^{\ell}|_t$ as the t-th bit of G_j^{ℓ}.
2. Compute the ℓ-th round message of P_i of the inner protocol π: $m_i^{\ell} = \mathsf{Nextmsg}_i\left(x_i, r_i, (M^{<\ell-1}, \bar{m}^{\ell-1})\right)$.
3. Set $d_{i,t}^{\ell} = \rho_{i,t,b}^{\ell}$ as the randomness used to generate P_i's Π_{OT} messages (acting as the receiver) corresponding to the bit $b = g_{i,t}$.
4. Generate the verification circuits $\mathrm{V}_{i,j}^{\ell}$ for all $j \in [N], t \in [T]$:

$$\mathrm{V}_{i,j}^{\ell}(\alpha_{j,t}) = \begin{cases} m_j^{\ell} = \mathsf{GC.Eval}(\widehat{\mathrm{C}}_j^{\ell}, \alpha_{j,t}) \\ \text{output } \mathsf{dataKeys}_{i,j}^{\ell+1}[m_j^{\ell}] \end{cases}$$

5. Generate input labels $\mathbf{vKeys}_{i,j}^{\ell} \leftarrow \mathsf{GC.Gen}(1^\lambda)$ and garble the circuit $\widehat{\mathrm{V}}_{i,j}^{\ell} \leftarrow \mathsf{GC.Garble}(\{\mathbf{vKeys}_{i,j}^{\ell}\}_j, \mathrm{V}_{i,j}^{\ell})$ for $j \in [N]$.
6. Generate P_i's Π_{OT} message (acting as the sender) corresponding to the Π_{OT} message of P_j (acting as the receiver) corresponding to the bit $b = g_{j,t}$ for all $j \in [N], t \in [T]$:

$$\nu_{i,j,t}^{\ell} = \mathsf{S}(\mathrm{crs}_{\mathsf{OT}}, \mu_{j,t,b}^{\ell}, \{\mathsf{vKeys}_{i,j,t'}^{\ell}\}_{t'})$$

   Since the input $\alpha_{j,t}$ to the *verification* circuit $\mathrm{V}_{i,j}$ is a κ-bit garbled label for the *commitment* garble circuit $\widehat{\mathrm{C}}_j$, each OT sender message $\nu_{i,j,t}$ includes κ pairs of labels vKeys. That said, $(t-1)\cdot\kappa < t' \le t\cdot\kappa$.
7. Select the input labels $\mathsf{stateKeys}_i^{\ell+1}[M^{<\ell-1}, \bar{m}^{\ell-1}]$ for the next round $(\ell+1)$, corresponding to the messages $M^{<\ell-1}, \bar{m}^{\ell-1}$.

Output: $(\mathsf{stateKeys}_i^{\ell+1}[M^{<\ell-1}, \bar{m}^{\ell-1}], \{\widehat{\mathrm{V}}_{i,j}^{\ell}, \nu_{i,j,k}^{\ell}, d_{i,k}^{\ell}\}_{j,k}, m_i^{\ell})$.

Fig. 7. Pseudocode of circuit F_i^{ℓ}

6.2 Security Proof

Theorem 6.1. *Let f be an arbitrary N-party functionality. Assume the existence of two-round adaptively secure malicious oblivious transfer protocol Π_{OT} in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model and an N-party malicious constant-round adaptively secure computation protocol π for f in $\mathcal{F}_{\mathsf{CRS}}$. Then the two-round protocol Π_{MPC}, presented in Protocol 1, UC-securely realizes the ideal functionality $\mathcal{F}_f$ in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model against adaptive corruption of any subset of the parties by a malicious adversary.*

The protocol π can be instantiated based on simulatable PKE [17] in the CRS model. In the semi-honest setting, no CRS is required and the protocol π can be instantiated based on augmented NCE [17]. See the full version for details.

The security proof is provided in the full version.

Acknowledgments. We thank the anonymous reviewers of TCC 2018 for their insightful comments. Huijia Lin was supported by NSF grants CNS-1528178, CNS-1514526, CNS-1652849 (CAREER), a Hellman Fellowship, the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236, and a subcontract No. 2017-002 through Galois. Antigoni Polychroniadou was supported by the Junior Simons Fellowship awarded by the Simons Society of Fellows. Muthuramakrishnan Venkitasubramaniam was supported by Google Faculty Research Grant and NSF Award CNS-1526377 and this work was partly carried out during a visit to DIMACS supported by the National Science Foundation under grant number CNS-1523467. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Removing erasures with explainable hash proof systems. In: Fehr, S. (ed.) PKC 2017, Part I. LNCS, vol. 10174, pp. 151–174. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_7
2. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_29
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC, pp. 1–10. ACM Press, May 1988
4. Benhamouda, F., Blazy, O., Ducas, L., Quach, W.: Hash proof systems over lattices revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 644–674. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_22
5. Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 500–532. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_17

6. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: improvements and extensions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16, pp. 1292–1303. ACM Press, October 2016
7. Boyle, E., Gilboa, N., Ishai, Y.: Group-based secure computation: optimizing rounds, communication, and computation. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 163–193. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_6
8. Boyle, E., Gilboa, N., Ishai, Y., Lin, H., Tessaro, S.: Foundations of homomorphic secret sharing. In: ITCS (2018, to appear)
9. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 190–213. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_8
10. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
11. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_4
12. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: 28th ACM STOC, pp. 639–648. ACM Press, May 1996
13. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_2
14. Canetti, R., Goldwasser, S., Poburinnaya, O.: Adaptively secure two-party computation from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 557–585. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_22
15. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, pp. 494–503. ACM Press, May 2002
16. Canetti, R., Poburinnaya, O., Venkitasubramaniam, M.: Better two-round adaptive multi-party computation. In: Fehr, S. (ed.) PKC 2017, Part II. LNCS, vol. 10175, pp. 396–427. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_14
17. Canetti, R., Poburinnaya, O., Venkitasubramaniam, M.: Equivocating yao: constant-round adaptively secure multiparty computation in the plain model. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th ACM STOC, pp. 497–509. ACM Press, June 2017
18. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_17
19. Choi, S.G., Katz, J., Wee, H., Zhou, H.-S.: Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 73–88. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_6
20. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 630–656. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_31

21. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
22. Dachman-Soled, D., Katz, J., Rao, V.: Adaptively secure, universally composable, multiparty computation in constant rounds. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 586–613. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_23
23. Damgård, I., Polychroniadou, A., Rao, V.: Adaptively secure multi-party computation from LWE (via equivocal FHE). In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016, Part II. LNCS, vol. 9615, pp. 208–233. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_9
24. Garay, J.A., Wichs, D., Zhou, H.-S.: Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 505–523. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_30
25. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_4
26. Garg, S., Polychroniadou, A.: Two-round adaptively secure MPC from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 614–637. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_24
27. Garg, S., Srinivasan, A.: Garbled protocols and two-round MPC from bilinear maps. In: 58th FOCS, pp. 588–599. IEEE Computer Society Press (2017)
28. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 468–499. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_16
29. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC, pp. 218–229. ACM Press, May 1987
30. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
31. Dov Gordon, S., Liu, F.-H., Shi, E.: Constant-round MPC with fairness and guarantee of output delivery. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 63–82. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_4
32. Halevi, S., Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. J. Cryptol. **25**(1), 158–193 (2012)
33. Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 637–653. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_37
34. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_26
35. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA, pp. 448–457. ACM-SIAM, January 2001

36. Peikert, C., Shiehian, S.: Multi-key FHE from LWE, revisited. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 217–238. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_9
37. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_31
38. Yao, A.C.C.: Theory and applications of trapdoor functions (extended abstract). In: 23rd FOCS, pp. 80–91. IEEE Computer Society Press, November 1982