

Two-Round Adaptively Secure MPC from Indistinguishability Obfuscation

Sanjam Garg¹ and Antigoni Polychroniadou^{2,*}

¹ University of California, Berkeley

`sanjamg@berkeley.edu`

² Aarhus University, Denmark

`antigoni@cs.au.dk`

Abstract. Adaptively secure Multi-Party Computation (MPC) first studied by Canetti, Feige, Goldreich, and Naor in 1996, is a fundamental notion in cryptography. Adaptive security is particularly hard to achieve in settings where arbitrary number of parties can be corrupted and honest parties are not trusted to properly erase their internal state. We did not know how to realize constant round protocols for this task even if we were to restrict ourselves to semi-honest adversaries and to the simpler two-party setting. Specifically the round complexity of known protocols grows with the depth of the circuit the parties are trying to compute. In this work, using indistinguishability obfuscation, we construct a UC two-round Multi-Party computation protocol secure against any active, adaptive adversary corrupting an arbitrary number of parties.

1 Introduction

The notion of *secure computation* is central in cryptography. Introduced in the seminal work of [41, 30] secure multiparty computation (MPC) allows several mutually distrustful parties P_1, \dots, P_n to compute a joint function f on their private inputs (x_1, \dots, x_n) , in a way that ensures that honest parties obtain the correct outputs and no group of colluding malicious parties learns anything beyond their own inputs and the prescribed output. For this problem, we are interested in the natural setting where the attacker can on-the-fly decide on which parties to corrupt. This model of *adaptive* corruption was first studied by Canetti et al. [9]. In this paper we consider adaptive adversaries that are allowed to corrupt *arbitrary number* of honest parties. Additionally we only consider *non-erasable* protocols, specifically the protocols whose security does not depend on having honest parties erase any of their internal state. We refer the reader to [9, Section 1] for discussion on the importance of considering adaptive adversaries.

* Research supported by the Danish National Research Foundation and the National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation and from the Center for Research in Foundations of Electronic Markets (CFEM), supported by the Danish Strategic Research Council. Also supported by ISF grant no. 1255/12.

One fundamental complexity measure of an MPC protocol is its *round complexity*. For the static setting, Yao’s original two-party secure computation protocol [41] was already round-optimal. Analogous results for the multi-party setting were obtained recently [1, 22].

However achieving similar results in the adaptive setting has remained open. In the case where all but one of the parties can be corrupted, [36, 34] and [35, 27] including the concurrent work of [18], propose constant round two-party and multi-party protocols, respectively. On the other hand, round complexity of all known adaptively secure protocols secure against an arbitrary number of corruptions grows (see, e.g. [14, 27, 16]) linearly in the depth of the circuit that the parties are trying to compute. We stress that for this problem, this limitation holds for essentially every special case of interest — namely, even if we were to restrict to semi-honest/passive adversaries or to the special case of two-party protocols. In this work we ask the following fundamental question:

Is it possible to construct a constant round protocol secure against adaptive corruption of arbitrary number of parties?

1.1 Our Result

We answer the above question in the affirmative and show how to obtain a two-round adaptively secure MPC protocol. Specifically:

Theorem 1 (informal). *Assuming sub-exponentially secure indistinguishability obfuscation and other standard assumptions, we show that arbitrary functions can be UC-securely [8] computed in the presence of adaptive, active corruption of arbitrary number of parties with just two rounds of broadcast messages.*

We stress that in the above claim we are in the setting where security holds against an adversary corrupting any arbitrary number of parties. Furthermore, honest parties in our case are not required to erase anything. Also note that our results are for the strongest notion of security, the UC security. This means that our protocol remains secure even when multiple instances of our protocol are executed simultaneously. Since it is impossible to achieve UC security for dishonest majority without assuming trusted setup assumptions [10, 12, 37], we base our construction in the common reference string model.

In our results we consider an asynchronous multi-party network³ where the communication is open (i.e. all the communication between the parties is seen by the adversary) and delivery of messages is not guaranteed. For simplicity, we assume that the delivered messages are authenticated. This can be achieved using standard methods.

³ The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

1.2 Independent Work

In two very recent concurrent and independent works, [15, 11] construct constant round protocols with security against a semi-honest, adaptive adversary corrupting any number of parties. Both works can obtain a constant round malicious version of their protocols by applying the [14] compiler.

In our paper we construct a two-round multi-party protocol with security against a malicious, adaptive adversary corrupting any number of parties. In contrast, the protocols of [15] and [11] require more rounds. Furthermore, the construction of [11] solves the problem only for the special case of two parties. Note that the reduction in our result and the result of [11] involves a sub-exponential loss of security.

Last but not least, our protocol and the protocol of [11] are also leakage tolerant. The semi-honest version of [11] is also incoercible with respect to one of the parties.

1.3 Technical Difficulties and New Ideas

The key technical tool that we use in our construction is obfuscation so let us start by recalling it briefly.

Obfuscation. Obfuscation was first rigorously defined and studied by Barak et al. [4]. Most famously, they defined the notion of *virtual black box (VBB)* obfuscation, and proved that this notion is impossible to realize in general — i.e., there exist functions, though a bit unnatural, that are VBB unobfuscatable.

Barak et al. also defined a weaker notion of *indistinguishability obfuscation ($i\mathcal{O}$)*, which avoids their impossibility results. Indistinguishability obfuscation requires that for any two circuits C_0, C_1 of similar size that *compute the same function*, it is hard to distinguish an obfuscation of C_0 from an obfuscation of C_1 . In a recent result, Garg et al. [23] proposed a construction of $i\mathcal{O}$ for all circuits, basing security on assumptions related to multilinear maps [21].

Starting point — Garg et al. [22] construction. In a recent work, Garg et al. [22] constructed a two-round multiparty computation protocol secure against static adversaries. Though our goal is to realize a protocol secure in the adaptive setting it would be illustrative to see how Garg et al.’s construction works.

With the goal of explaining intuition [22] better we will describe the ideas assuming we have access to VBB obfuscation, rather than just indistinguishability obfuscation. We start by noting that two rounds of interaction are essential for realizing multiparty secure computation. This is because a 1-round protocol is inherently susceptible to the “residual function” attack in which a corrupted party could repeatedly evaluate the “residual function” with the inputs of the honest parties fixed on many different inputs of its own (e.g., see [33]). This attack can be circumvented by having two rounds of interaction — where in the first round the parties commit to their inputs and the output is generated

only in the second round. The first round commitments help guarantee that the “residual function” attack can not be performed in this setting.

The key idea of the Garg et al. construction is to have every party commit to its input along with its randomness in the first round. The second round of the Garg et al. protocol is actually a simple compiler: it takes any (possibly highly interactive) underlying MPC protocol, and has each party obfuscate their “next-message” function in that protocol, providing one obfuscation for each round. This enables each party to independently evaluate the obfuscations one by one, generating messages of the underlying MPC protocol and finally obtain the output. Party i ’s next-message circuit for round j in the underlying MPC protocol depends on its input x_i and randomness r_i (which are hard-coded in the obfuscation). This circuit takes as input the transcript through round $j - 1$, and it produces as output the next broadcast message.

However, there is another complication. Unlike the initial interactive protocol being compiled, the obfuscations are susceptible to a “reset” attack – i.e., they can be evaluated on multiple inputs. To prevent such an attack, we need to limit the obfuscations to be used for evaluation only on a unique set of values – namely, values consistent with the inputs and randomness that the parties committed to in the first round, and the current transcript of the underlying MPC protocol. Note that this would implicitly fix the transcript to a unique value. To ensure this consistency, Garg et al. [22] use non-interactive zero-knowledge (NIZK) proofs. Since the NIZKs apply not only to the committed values of the first round, but also to the transcript as it develops in the second round, the obfuscations themselves must also generate these NIZKs “on the fly”. In other words, the obfuscations are now augmented to perform not only the next-message function, but also to prove that their output is consistent with their input, randomness and transcript so far. Also, obfuscations in round j of the underlying MPC protocol verify NIZKs associated to obfuscations in previous rounds before providing any output.

Garg et al. show that this construction can be adapted so that security can be based on indistinguishability obfuscation alone but we will not delve into that. Instead we will argue that this approach is fundamentally problematic for achieving the task at hand.

Our approach – starting afresh. Note that the above intuitive description uses multiple obfuscations that are generated by honest parties. This however only works in the static setting and our goal is adaptive security. The challenge in proving adaptive security is that, a simulator would have a hard time explaining these obfuscations as being honestly generated. Towards solving this problem we first would like to limit the use of obfuscation in our construction; specifically not requiring honest parties to generate any obfuscations.

Still assuming we have access to VBB obfuscation, we need a fresh direction to solve the above problem. Here is our first stab at the problem: assume the parties had access to a trusted third party. In this case each party could encrypt its input and deliver it to the trusted party. The trusted party could then decrypts these values to obtain the inputs of all the parties, compute the function on the inputs

and then deliver the output back to the parties. Our idea is to have an obfuscated program given out as part of the CRS implement this trusted party. Just like the Garg et al. construction, in order to make this construction secure against “residual function” attack we will need to consider a setting with two rounds. In the first round, we will have all parties commit to their inputs and then in the second round we will have them provide encryptions of the openings previously committed.

Making this construction adaptively secure seems more amenable — specifically, by using adaptive commitments for the first round and a deniable encryption scheme for the second. We actually need the first round commitments to be simulation extractable. This allows our simulator to extract the values committed to by the adversary on behalf of corrupted parties, even as it equivocates on its own commitments. Once the simulator has access to the inputs of the corrupted outputs it can force the output by including it in its own second round encryption.

Basing it on Indistinguishability Obfuscation. The protocol described so far relies on VBB and we would like to instantiate our construction based on $i\mathcal{O}$. The CRS of the scheme includes an obfuscation that takes as input encryptions of inputs of all the parties and computes the desired functionality on their decryptions. A reader might have observed that this bears resemblance with functional encryption or even multi-input functional encryption [31]. One might wonder if the use of “two key trick” can help us realize this construction using just indistinguishability obfuscation — in a way similar to the functional encryption construction of Garg et al. [23]. In particular the idea would be that each party encrypts its input along with the opening twice under two different keys and attach along with them a NIZK proof proving that they indeed encrypt the same value.

Unfortunately, this solution is fundamentally problematic as we are in the adaptive setting. Even if we were to use an adaptively secure NIZK the problem is that NIZKs given on deniable encryptions are useless. This is because the encryption scheme is deniable. The deniability of the encryption scheme allows the adversary to encrypt two different plaintexts under the two public keys but still succeed in explaining them as encrypting the same message. This also allows the attacker to successfully prove that the two ciphertexts indeed encrypt the same message.

In summary, what we really need is a system with two ciphertexts and a proof proving that the two ciphertexts encrypt the same message with the property that only valid proofs exists. Additionally we need the property that both the ciphertexts and the proof can be denied upon in the proof of security. These requirements indeed seem to be in conflict with each other. For example, simultaneously achieving perfect soundness for NIZK and the ability to explain the simulated proofs as though they were honestly generated seems like a bottleneck.

Our solution to this seemingly paradoxical problem is to first construct a core two key encryption scheme which comes attached with a NIZK and then build deniability on top of it. In particular, the underlying core encryption scheme con-

sists of two copies of a perfectly correct encryption scheme along with a NIZK proving that the two ciphertexts encrypt the same message and it is combined with a language which also binds it with the commitments of the first round. The NIZK we use will have statistical soundness. This underlying encryption scheme is then made *deniable* using the Sahai and Waters [40] Universal Deniable Encryption (UDE) transformation. Briefly, UDE takes *any* encryption scheme and converts it to *deniable* so that ciphertexts are still indistinguishable from the usual ciphertexts of the underlying core encryption. Hence, our resulting encryption is deniable in a very strong sense — specifically, it allows the encryptor to deny not just on the two ciphertexts but also on the NIZK directly. However, interestingly proofs for invalid statements do not exist.

Finally various other technical challenges arise in the security proof. For example, in the proof of security the simulator needs to hardcode the output that the adversary gets as part of its ciphertext in a way that remains indistinguishable from an honest execution. In order to force the output, the core encryption scheme which is plugged into the UDE transformation is combined with the language which implicitly includes a trapdoor mode. In its trapdoor mode, the simulator can in particular plant the output of the function inside the encryptions it generates on behalf of honest parties. Then the obfuscation checks for such a trapdoor and acts accordingly. We refer the reader to the full construction and proof for details on how we resolve this and other issues.

1.4 Application to Leakage Tolerant Protocols

As another application of our techniques, we observe that our adaptively secure protocol is also leakage tolerant in a way that previous protocols failed to be. The study of leakage tolerant protocols was initiated by Bitansky et al. [5] and Garg et al. [25]. Very roughly, leakage tolerant protocols preserve security even when the adversary can obtain arbitrary leakage on the entire internal state of honest parties, however only up to the leaked information.

One limitation of known leakage tolerant secure computation protocols [7] (see also [17]) from the literature is that the leakage in the ideal world queries needs to depend on the inputs of all honest parties rather than just on the input of the party being leaked upon. Our adaptively secure protocol also turns out to be leakage resilient further avoiding this limitation. Another advantage of our protocol is that it is much simpler than the Boyle et al. [7] construction.

In a recent result, Garg et al. [24] show an alternative way of avoiding this limitation, without using indistinguishability obfuscation. However their result is restricted to a setting where at least one of the parties is never leaked on. We do not make such an assumption.

2 Preliminaries

In this section we recall preliminary notions needed in this work. We will start by recalling notions of indistinguishability obfuscation and non-interactive zero-

knowledge. Next we recall the notion of publicly deniable encryption scheme that we adapt from [40].

2.1 Notation

Throughout the paper $\lambda \in \mathbb{N}$ will denote the security parameter. We say that a function $f : \mathbb{N} \rightarrow \mathbb{R}$ 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 $\{1, \dots, n\}$. 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 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 $\mathcal{D}_1 \equiv \mathcal{D}_2$.

2.2 Indistinguishability Obfuscators

We will start by recalling the notion of indistinguishability obfuscation ($i\mathcal{O}$) recently realized in [23] using candidate multilinear maps [21].

Definition 1 (Indistinguishability Obfuscator ($i\mathcal{O}$)). *A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:*

- For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x , we have that

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1$$

- For any (not necessarily uniform) PPT distinguisher D , there exists a negligible function α such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x , then

$$\left| \Pr [D(i\mathcal{O}(\lambda, C_0)) = 1] - \Pr [D(i\mathcal{O}(\lambda, C_1)) = 1] \right| \leq \text{negl}(\lambda)$$

2.3 Non-Interactive Zero-Knowledge Proofs

Let \mathcal{R} be an NP-relation. For pairs $(x, w) \in \mathcal{R}$ we call x the statement and w the witness. Let \mathcal{L} be the language consisting of statements in \mathcal{R} . A Non-Interactive Zero Knowledge (NIZK) Proof system [6, 19] consists of three PPT algorithms (K, P, V) , a common reference string generation algorithm K , a prover P and a verifier V .

- $K(1^\lambda)$ expects as input the unary representation of the security parameter λ and outputs a common reference string σ of length $\Omega(\lambda)$.
- $P(\sigma, x, w)$ takes as input a common reference string σ , a statement x together with a witness w such that $\mathcal{R}(x, w)$ and produces a proof π .

- $V(\sigma, x, \pi)$ takes as input a common reference string σ , a statement x , a proof π and outputs 1 if the proof is accepting and 0 otherwise.

We call (K, P, V) a non-interactive proof system for \mathcal{R} if it satisfies the following properties.

PERFECT COMPLETENESS. A proof system is complete if an honest prover with a valid witness can convince an honest verifier. Formally, $\forall x \in \mathcal{L}, \forall w$ witness of x

$$\Pr \left[\sigma \leftarrow K(1^\lambda); \pi \leftarrow P(\sigma, x, w) : V(\sigma, x, \pi) = 1 \right] = 1.$$

STATISTICAL SOUNDNESS. A proof system is sound if it is infeasible to convince an honest verifier when the statement is false. Formally, we have

$$\Pr \left[\sigma \leftarrow K(1^\lambda); \exists(x, \pi) : x \notin \mathcal{L} \wedge V(\sigma, x, \pi) = 1 \right] < \text{negl}(\lambda).$$

COMPUTATIONAL ZERO-KNOWLEDGE. We say a non-interactive proof (K, P, V) is computational zero-knowledge if there exists a PPT simulator $S = (S_1, S_2)$, where S_1 returns a simulated common reference string σ together with a simulation trapdoor τ that enables S_2 to simulate proofs without having access to the witness. For all non-uniform PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following quantity is upper bounded by a negligible function:

$$\left| \Pr \left[\sigma \leftarrow K(1^\lambda); (x, \text{state}) \leftarrow \mathcal{A}_1(\sigma); \pi \leftarrow P(\sigma, x, w) : \mathcal{A}_2(x, \sigma, \pi, \text{state}) = 1 \right] - \Pr \left[(\sigma, \tau) \leftarrow S_1(1^\lambda); (x, \text{state}) \leftarrow \mathcal{A}_1(\sigma); \pi \leftarrow S_2(\sigma, \tau, x) : \mathcal{A}_2(x, \sigma, \pi, \text{state}) = 1 \right] \right|.$$

2.4 Double Key Encryption and its Deniable Variant

Our protocol will use a special publicly deniable encryption scheme that we construct by first describing a core public-key encryption scheme that we then transform to its deniable variant using the Universal Deniable Encryption (UDE) transformation of [40].

Let $(\text{Setup}, \text{Enc}, \text{Dec})$ be a perfectly correct IND-CPA secure public-key encryption scheme and let (K, P, V) be a NIZK proof system with statistical soundness and computational zero-knowledge. The core encryption scheme we consider is very similar to the Naor-Yung CCA [39] secure encryption scheme. Recall that in the Naor-Yung construction a ciphertext consists of encryptions of a message under two different public keys and a NIZK proof certifying that the two ciphertexts indeed encrypt the same message. In our encryption scheme a ciphertext will also consist of two ciphertexts under the two public keys but the NIZK proof will be used to certify a more sophisticated requirement. More formally:

Definition 2 (Double Key Encryption Scheme). Let $(\text{Setup}, \text{Enc}, \text{Dec})$ be a IND-CPA secure encryption scheme with perfect correctness. Let (K, P, V) be a NIZK proof system for an NP-Language \mathcal{L} . A Double Key encryption scheme, parametrised by a language \mathcal{L} , consists of three algorithms $\text{DK}_{\mathcal{L}} = (\text{Setup}_{\text{DK}}, \text{Enc}_{\text{DK}}, \text{Dec}_{\text{DK}})$.

- $\text{Setup}_{\text{DK}}(1^\lambda, 1^\ell)$ is a polynomial time procedure that takes as input the unary representation of the security parameter λ , the description of length of messages encrypted 1^ℓ . It computes $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \text{Setup}(1^\lambda)$ and the common reference string $\sigma \leftarrow K(1^\lambda)$ for the NIZK proof. It outputs the public key $PK = (pk_0, pk_1, \sigma)$ and the secret key $SK = (sk_0, sk_1)$.
- $\text{Enc}_{\text{DK}}(PK, m_0, m_1, \text{aux}, w; r)$: This polynomial time procedure takes as input public key $PK = (pk_0, pk_1, \sigma)$, messages $m_0, m_1 \in \{0, 1\}^\ell$, auxiliary information aux and some w which will be used as part of the witness for the language \mathcal{L} . It generates $c = \text{Enc}(pk_0, m_0; s_0)$ and $c' = \text{Enc}(pk_1, m_1; s_1)$ and outputs (c, c', π) , where $\pi \leftarrow P(\sigma, (c, c', \text{aux}), (m_0, m_1, s_0, s_1, w))$ for the language \mathcal{L} .
- $\text{Dec}_{\text{DK}}(PK, SK, (c, c', \pi), \text{aux})$: is a polynomial time procedure that takes as input $PK = (pk_0, pk_1, \sigma)$, $SK = (sk_0, sk_1)$, ciphertext (c, c', π) and auxiliary information aux . Outputs \perp , in case that $V(\sigma, (c, c', \text{aux}), \pi) = 0$ else output $(\text{Dec}(sk_0, c), \text{Dec}(sk_1, c'))$.

Double Key Deniable Encryption Scheme. Next we want to transform the above core public key encryption into its deniable variant using the UDE transformation of Sahai and Waters [40, Section 4.2]. In particular, once we plug the above $\text{DK}_{\mathcal{L}}$ double key encryption scheme in the UDE transformation, we obtain a double key *deniable* encryption scheme $\text{DDK}_{\mathcal{L}} = (\text{Setup}_{\text{DDK}}, \text{Enc}_{\text{DDK}}, \text{Dec}_{\text{DDK}}, \text{DenEnc}_{\text{DDK}}, \text{Explain}_{\text{DDK}})$ parametrized by the language \mathcal{L} with associate relation $\mathcal{R}_{\mathcal{L}}$ where the procedures Enc_{DDK} and Dec_{DDK} are same as Enc_{DK} and Dec_{DK} . Here $\text{Setup}_{\text{DDK}}$ is obtained by augmenting the procedure Setup_{DK} to additionally output a public denying key DK generated using $\text{UniversalSetup}(PK)$ as defined in [40, Section 4.2] which is going to be included in PK . Further the scheme is augmented with the following two procedures where $PK = (\sigma, pk_0, pk_1, DK)$.

- $\text{DenEnc}_{\text{DDK}}(PK, m_0, m_1, \text{aux}, w; r)$ is a polynomial time procedure that takes as input PK which includes the public denying key DK , $m_0, m_1 \in \{0, 1\}^\ell$, auxiliary information aux and witness w and uses random coins r . It then outputs (c, c', π) .
- $\text{Explain}_{\text{DDK}}(PK, (c, c', \pi), (m_0, m_1, \text{aux}, w); u)$: This polynomial time procedure takes as input public key PK which includes the public denying key DK , messages $m_0, m_1 \in \{0, 1\}^\ell$, auxiliary information aux and witness w . It also takes as input a value (c, c', π) and outputs a string e , that is the same size as the randomness r taken by $\text{DenEnc}_{\text{DDK}}$ above.

This new scheme has the following two additional properties.

Indistinguishability of source of ciphertext. We say that the scheme has indistinguishability of source of ciphertext if for any λ and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following quantity can be upper bounded by a negligible function:

$$\left| \Pr \left[\begin{array}{l} (PK, SK) \leftarrow \text{Setup}_{\text{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \text{aux}, w) \leftarrow \mathcal{A}_1(PK), \\ ct = \text{Enc}_{\text{DDK}}(PK, m_0, m_1, \text{aux}, w; r) \\ \mathcal{A}_2(PK, ct) = 1 \end{array} \right] - \Pr \left[\begin{array}{l} (PK, SK) \leftarrow \text{Setup}_{\text{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \text{aux}, w) \leftarrow \mathcal{A}_1(PK), \\ ct = \text{DenEnc}_{\text{DDK}}(PK, m_0, m_1, \text{aux}, w; r) \\ \mathcal{A}_2(PK, ct) = 1 \end{array} \right] \right|$$

Indistinguishability of explanation. We say that the scheme has indistinguishability of explanation if for any λ and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following quantity can be upper bounded by a negligible function:

$$\left| \Pr \left[\begin{array}{l} (PK, SK) \leftarrow \text{Setup}_{\text{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \text{aux}, w) \leftarrow \mathcal{A}_1(PK), \\ ct = \text{DenEnc}_{\text{DDK}}(PK, m_0, m_1, \text{aux}, w; r) \\ \mathcal{A}_2(PK, ct, r) = 1 \end{array} \right] - \Pr \left[\begin{array}{l} (PK, SK) \leftarrow \text{Setup}_{\text{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \text{aux}, w) \leftarrow \mathcal{A}_1(PK), \\ ct = \text{DenEnc}_{\text{DDK}}(PK, m_0, m_1, \text{aux}, w; r) \\ e = \text{Explain}_{\text{DDK}}(PK, ct, (m_0, m_1, \text{aux}, w); u) \\ \mathcal{A}_2(PK, ct, e) = 1 \end{array} \right] \right|$$

We remark that the [40] deniable encryption scheme immediately implies a deniable encryption scheme for multi-bit messages of any polynomial length k bits by creating a ciphertext for a k -bit message as a sequence of k single bit encryptions. Our construction cannot support the above bit-by-bit encryption since every single encryption takes longer messages. However the Sahai-Waters construction is selectively secure and the security can be amplified to the adaptive setting (as defined above) at the cost of a sub-exponential loss in the security. In other words we can realize the above definition assuming sub-exponential hardness on the assumptions made by Sahai-Waters.

2.5 Equivocal and Extractable Commitments

An Equivocal and Extractable Commitment scheme COM consists of a tuple of PPT algorithms $(\text{Setup}_{\text{Com}}^{\text{bind}}, \text{Setup}_{\text{Com}}^{\text{equiv}}, \text{Com}, \text{Extr}, \text{Equiv})$. We will describe our definitions for the setting of bit commitment and note that they extend to the setting of strings in a natural way.

- $\text{Setup}_{\text{Com}}^{\text{bind}}(1^\lambda)$ expects as input the unary representation of the security parameter λ and outputs a public parameter CK together with a trapdoor μ (used for extraction).
- $\text{Setup}_{\text{Com}}^{\text{equiv}}(1^\lambda)$ expects as input the unary representation of the security parameter λ and outputs a public parameter CK together with trapdoors μ and ν (used for extraction and equivocation).
- $\text{Com}(CK, b; r)$ takes as input CK , a bit $b \in \{0, 1\}$ and randomness $r \in \{0, 1\}^\lambda$ and outputs a commitment β .

Let us define the following language (the extraction procedure Extr is defined below):

$$\mathcal{L}_{\text{Com}} = \{(\beta, b) \mid \exists t : \beta = \text{Com}(CK, b; t) \vee b = \text{Extr}(CK, t, \beta)\}.$$

We note that the language naturally extends to a setting where commitments are defined over strings instead of just bits. Also we defined associated relation \mathcal{R}_{Com} . The above commitment scheme should satisfy the following properties.

INDISTINGUISHABILITY OF PUBLIC PARAMETERS. We require that:

$$\left| \Pr \left[(CK, \mu) \leftarrow \text{Setup}_{\text{Com}}^{\text{bind}}(1^\lambda) : \mathcal{A}(CK, \mu) = 1 \right] - \Pr \left[(CK, \mu, \nu) \leftarrow \text{Setup}_{\text{Com}}^{\text{equiv}}(1^\lambda) : \mathcal{A}(CK, \mu) = 1 \right] \right| < \text{negl}(\lambda).$$

COMPUTATIONAL HIDING. Hiding means that no computationally bounded adversary can distinguish as to which bit is locked in the commitment. Let \mathcal{A} be any non-uniform adversary running in time $\text{poly}(\lambda)$. We say that the commitment scheme is computationally hiding if:

$$\Pr \left[b = b' \mid \begin{array}{l} b \leftarrow \{0, 1\}; (CK, \mu) \leftarrow \text{Setup}_{\text{Com}}^{\text{bind}}(1^\lambda); \\ \beta = \text{Com}(CK, b; r); b' \leftarrow \mathcal{A}(\beta) \end{array} \right] = \frac{1}{2} + \text{negl}(\lambda).$$

The same applies to the setup algorithm $\text{Setup}_{\text{Com}}^{\text{equiv}}$.

PERFECTLY BINDING. Intuitively speaking, binding requires that no (even unbounded) adversary can open the commitment in two different ways. Here, we define the strongest variant known as *perfectly binding*. Formally we require that for all $(CK, \mu) \leftarrow \text{Setup}_{\text{Com}}^{\text{bind}}(1^\lambda)$ there exists no values (r_0, r_1) such that $\text{Com}(CK, 0; r_0) = \text{Com}(CK, 1; r_1)$. For perfectly binding we require that either $(c, 0) \in \mathcal{L}_{\text{Com}}$ or $(c, 1) \in \mathcal{L}_{\text{Com}}$, but not both.

POLYNOMIAL EQUIVOCALITY. The setup algorithm $\text{Setup}_{\text{Com}}^{\text{equiv}}$ generates public parameters together with trapdoors μ and ν such that Equiv using ν is able to produce polynomially many fake commitments, using the same CK , which can then be explained to either 0 and 1. More formally, Equiv can be viewed as a pair of PPT algorithms $(\text{Equiv}_1, \text{Equiv}_2)$ such that the following holds. Let $(CK, \mu, \nu) \leftarrow \text{Setup}_{\text{Com}}^{\text{equiv}}(1^\lambda)$ then $(\beta, \text{state}) \leftarrow \text{Equiv}_1(CK, \nu)$ and $r_b \leftarrow \text{Equiv}_2(CK, \nu, \beta, \text{state}, b)$ such that $\text{Com}(CK, b; r_b) = \beta$. Furthermore we require that for $b \in \{0, 1\}$ the distribution of $\{(CK, \beta, r_b)\}$ generated in this way is computationally indistinguishable from the distribution $\{(CK, \beta, r_b)\}$ where $\beta = \text{Com}(CK, b; r_b)$.

SIMULATION EXTRACTABILITY. We require that the commitment remains binding for any adversary \mathcal{A} , even after \mathcal{A} obtains polynomially many equivocal commitments generated by Equiv along with their openings. More formally, the following quantity is negligible:

$$\Pr \left[b \neq b' \mid (CK, \mu, \nu) \leftarrow \text{Setup}_{\text{Com}}^{\text{equiv}}(1^\lambda); (\beta, b, r) \leftarrow \mathcal{A}^{\text{Equiv}^*(CK, \nu)}(CK); \right. \\ \left. \text{Com}((CK, b, r) = \beta \wedge \text{Extr}(CK, \mu, \beta) = b' \right]$$

where Equiv^* is either invoked as Equiv_1 without revealing the *state*, or as Equiv_2 which only expects as input fake commitments generated by previous invocations of Equiv_1 .

In this paper, we use the non-interactive equivocal and extractable commitment scheme of [14] (CLOS commitment) which assumes the existence of enhanced trapdoor permutations. At the heart of their commitment scheme is the Feige-Shamir trapdoor commitment scheme [20], which they transform to obtain a UC Commitment scheme secure against adaptive adversaries.

3 Our Protocol

In this section we will present our adaptively secure two-round MPC protocol, described in Figure 1. For simplicity, we assume that the delivered messages are authenticated. Also for simplicity of exposition, in the sequel, we will assume that random coins are an implicit input to the commitment and encryption functions, unless specified explicitly.

Theorem 2. *Let f be any deterministic poly-time function with n inputs and single output. Assume the existence of an Indistinguishability Obfuscator $i\mathcal{O}$, a Double Key Deniable encryption scheme $\text{DDK}_{\mathcal{L}} = (\text{Setup}_{\text{DDK}}, \text{Enc}_{\text{DDK}}, \text{Dec}_{\text{DDK}}, \text{DenEnc}_{\text{DDK}}, \text{Explain}_{\text{DDK}})$ and an adaptively secure Commitment scheme $\text{COM} = (\text{Setup}_{\text{Com}}^{\text{bind}}, \text{Setup}_{\text{Com}}^{\text{equiv}}, \text{Com}, \text{Extr}, \text{Equiv})$. Then the protocol Π presented in Figure 1 UC-securely realizes the ideal functionality \mathcal{F}_f in the \mathcal{F}_{CRS} -hybrid model with computational security against any adaptive, active adversary corrupting an arbitrary number of parties in two rounds of broadcast.*

Corollary 1. *Assume the existence of a sub-exponentially secure indistinguishability obfuscation and doubly enhanced trapdoor permutation then any ideal functionality \mathcal{F}_f can be UC-securely realized in the \mathcal{F}_{CRS} -model against any adaptive, active adversary corrupting an arbitrary number of parties. Furthermore this protocol involves only two rounds of broadcast.*

We start by noting that the protocol is correct. Observe that if all the parties behave honestly then the protocol ends us executing the circuit f on the inputs of all parties, leading to the correct output. Security is proved via a simulator provided in Section 4 and indistinguishability is argued in Section 5.

3.1 Extensions

Now we give some natural extensions of our protocol and remove assumptions that were made to simplify exposition.

Protocol *II*

Protocol *II* uses an Indistinguishability Obfuscator $i\mathcal{O}$, a Double Key Deniable encryption scheme $\text{DDK}_{\mathcal{L}} = (\text{Setup}_{\text{DDK}}, \text{Enc}_{\text{DDK}}, \text{Dec}_{\text{DDK}}, \text{DenEnc}_{\text{DDK}}, \text{Explain}_{\text{DDK}})$ based on the scheme $(\text{Setup}, \text{Enc}, \text{Dec})$ with perfect correctness, where the relation \mathcal{L} is defined below, and an adaptively secure Commitment scheme $\text{COM} = (\text{Setup}_{\text{Com}}^{\text{bind}}, \text{Com})$.^a Let $f : (\{0, 1\}^{\ell_{in}})^n \rightarrow \{0, 1\}^{\ell_{out}}$ be the circuit parties want to evaluate on their private inputs.

Private Inputs: Party P_i for $i \in [n]$, receives its input x_i .

CRS: Output (PK, CK, oP) as the common reference string generated as follows:

- Generate $(PK, SK) \leftarrow \text{Setup}_{\text{DDK}}(1^\lambda, 1^{\ell_{in} + \ell_{out}})$ where $PK = (\sigma, pk_0, pk_1, DK)$ and $SK = (sk_0, sk_1)$
- Generate $(CK, \mu) \leftarrow \text{Setup}_{\text{Com}}^{\text{bind}}(1^\lambda)$.
- Let $\text{oP} = i\mathcal{O}_{\text{Prog}_{sk_0, PK, CK, f}}$ be the obfuscation of the program $\text{Prog}_{sk_0, PK, CK, f}$, described in Figure 2.

Round 1: Each party P_i generates $\beta_i = \text{Com}(CK, x_i; \omega_i)$ and broadcasts it to all parties.

Round 2: Each party P_i generates $(c_i, c'_i, \pi_i) = \text{DenEnc}_{\text{DDK}}(PK, x_i || \phi^{\ell_{out}}, x_i || \phi^{\ell_{out}}, (i, \{\beta_j\}_{j \in [n]}), (0^{n \cdot \ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j \in [n]}; r_i))$ where ϕ is a special fixed symbol and $t_i = \omega_i$ and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. It then broadcasts (c_i, c'_i, π_i) to all parties.

Output phase: Each party P_i outputs $\text{oP}(\{\beta_j\}_{j \in [n]}, \{c_j, c'_j, \pi_j\}_{j \in [n]})$.

Language \mathcal{L} for the Double Key deniable encryption scheme $\text{DDK}_{\mathcal{L}}$:

Recall \mathcal{L}_{Com} as the language defined in Section 2.5, and let \mathcal{R}_{Com} be the associated relation. We have that $(c, c', (i, \{\beta_j\}_{j \in [n]})) \in \mathcal{L}$ if $(c, c', (i, \{\beta_j\}_{j \in [n]})) \in \mathcal{L}_1 \vee \mathcal{L}_2$ defined as follows:^b

$$\mathcal{L}_1 = \left\{ (c, c', (i, \{\beta_j\}_{j \in [n]})) \left| \begin{array}{l} \exists (m_0, m_1, s_0, s_1, (\{x_j\}_{j \in [n]}, \text{out}, \{t_j\}_{j \in [n]})) \text{ s.t.} \\ c = \text{Enc}(pk_0, m_0; s_0) \wedge c' = \text{Enc}(pk_1, m_1; s_1) \\ \wedge m_0 = m_1 = x_i || \phi^{\ell_{out}} \\ \wedge \mathcal{R}_{\text{Com}}((\beta_i, x_i), t_i) \end{array} \right. \right\}$$

$$\mathcal{L}_2 = \left\{ (c, c', (i, \{\beta_j\}_{j \in [n]})) \left| \begin{array}{l} \exists (m_0, m_1, s_0, s_1, (\{x_j\}_{j \in [n]}, \text{out}, \{t_j\}_{j \in [n]})) \text{ s.t.} \\ c = \text{Enc}(pk_0, m_0; s_0) \wedge c' = \text{Enc}(pk_1, m_1; s_1) \\ \wedge m_0 = x_i || \phi^{\ell_{out}} \wedge m_1 = \phi^{\ell_{in}} || \text{out} \\ \wedge \forall j \in [n], \mathcal{R}_{\text{Com}}((\beta_j, x_j), t_j) \\ \wedge \text{out} = f(\{x_j\}_{j \in [n]}) \end{array} \right. \right\}$$

^a We note that COM provides more procedures but we note that they only affect the proof. Hence for simplicity of exposition we skip mentioning them here.

^b Changes in \mathcal{L}_2 from \mathcal{L}_1 are highlighted in red.

Fig. 1. The *II* Protocol.

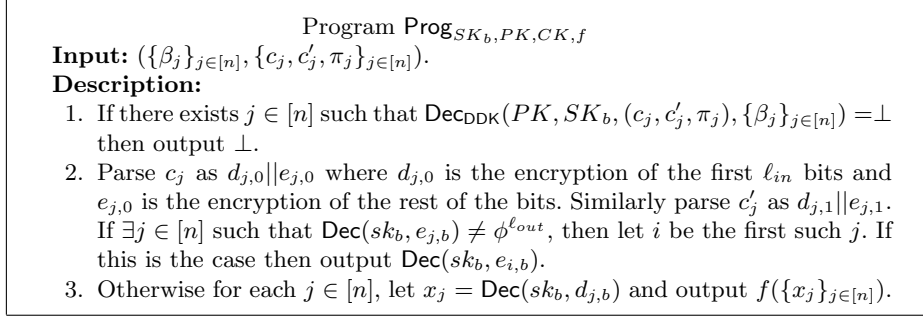


Fig. 2. The Program $\text{Prog}_{SK_b, PK, CK, f}$.

General Functionality. Our basic MPC protocol as described in Figure 1, only considers deterministic functionalities where all the parties receive the same output. We would like to generalize it to handle randomized functionalities and individual outputs (just as in [2]). First, the standard transformation from a randomized functionality to a deterministic one (See [29, Section 7.3]) works for this case as well. In this transformation, instead of computing some randomized function $g(x_1, \dots, x_n; r)$, the parties compute the deterministic function $f((r_1, x_1), \dots, (r_n, x_n)) \stackrel{\text{def}}{=} g(x_1, \dots, x_n; \oplus_{i=1}^n r_i)$. We note that this computation does not add any additional rounds. We note that since we are in the setting of adaptive security we can only realize *adaptively well-formed* [14] functionalities, which reveals its randomness if all the parties are corrupted.

Next, we move to individual outputs. Again, we use a standard transformation (See [38], for example). Given a function $g(x_1, \dots, x_n) \rightarrow (y_1, \dots, y_n)$, the parties can evaluate the following function which has a single output:

$$f((k_1, x_1), \dots, (k_n, x_n)) = (g_1(x_1, \dots, x_n) \oplus k_1 || \dots || g_n(x_1, \dots, x_n) \oplus k_n)$$

where g_i indicates the i^{th} output of g , and k_i is randomly chosen by the i^{th} party. Then, the parties can evaluate f , which is a single output functionality, instead of g . Subsequently every party P_i uses its secret input k_i to recover its own output. The only difference is that f has one additional exclusive-or gate for every circuit-output wire. Again, this transformation does not add any additional rounds of interaction.

Making CRS independent of the circuit being computed. Note that in our construction the obfuscation oP that is given as part of the CRS depends on the circuit f parties are trying to compute on their joint inputs. We can remove this dependence by using a universal circuit. Then the parties can feed in the universal circuit the actual circuit that they want along with their private inputs. However, the CRS will still depend on the size of the circuit. This is also the case for the protocols in [11, 15]. We can avoid this by setting a priori bound on the size of the circuit being computed. It would be interesting to remove the dependence of the CRS on the size of the circuit.

4 Description of our Simulator

Let \mathcal{A} be an active, adaptive adversary that interacts with parties running the protocol Π from Figure 1 in the \mathcal{F}_{CRS} -hybrid model. We construct a simulator \mathcal{S} (the ideal world adversary) with access to the ideal functionality \mathcal{F}_f , which simulates a real execution of Π with \mathcal{A} such that no environment \mathcal{Z} can distinguish the ideal world experiment with \mathcal{S} and \mathcal{F}_f from a real execution of Π with \mathcal{A} .

Recall that \mathcal{S} interacts with the ideal functionality \mathcal{F}_f and with the environment \mathcal{Z} . The ideal adversary \mathcal{S} starts by invoking a copy of \mathcal{A} and running a simulated interaction of \mathcal{A} with the environment \mathcal{Z} and the parties running the protocol. Our simulator \mathcal{S} proceeds as follows:

Simulated CRS: The common reference string is chosen by \mathcal{S} in the following manner (recall that \mathcal{S} chooses the CRS for the simulated \mathcal{A} as we are in the \mathcal{F}_{CRS} -hybrid model):

1. \mathcal{S} runs the setup algorithm $\text{Setup}_{\text{DDK}}(1^\lambda, 1^{\ell_{in} + \ell_{out}})$ of the Double Key deniable encryption scheme, but replaces its internal call to the algorithm K with $S = (S_1, S_2)$ of the NIZK proof system. More specifically, \mathcal{S} generates $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \text{Setup}(1^\lambda)$, $(\sigma, \tau) \leftarrow S_1(1^\lambda)$, along with the public denying key DK . It sets the public key $PK = (pk_0, pk_1, \sigma, DK)$.
2. \mathcal{S} runs the algorithm $\text{Setup}_{\text{Com}}^{\text{equiv}}(1^\lambda)$ of the adaptively secure commitment scheme COM and obtains (CK, μ, ν) .
3. \mathcal{S} computes $\text{oP} = i\mathcal{O}_{\text{Prog}_{sk_1, PK, CK, f}}$ where the latter is the obfuscation of the program Prog , as described in Figure 2, parameterized with the key sk_1 .

\mathcal{S} sets the common reference string equal to (PK, CK, oP) and locally stores (SK, τ, μ, ν) .

Looking ahead, the trapdoor μ will be used to extract the inputs of the corrupted parties and ν to equivocate on the commitment \mathcal{S} provides on behalf of honest parties. The trapdoor τ for the simulated σ will be used to generate simulated proofs.

Simulating the communication with \mathcal{Z} : Every input value that \mathcal{S} receives from \mathcal{Z} is written on \mathcal{A} 's input tape. Similarly, every output value written by \mathcal{A} on its own output tape is directly copied to the output tape of \mathcal{S} .

Simulating actual protocol messages in Π : Note that there might be multiple sessions executing concurrently. Let sid be the session identifier for one specific session. We will specify the simulation strategy corresponding to this specific session. The simulator strategy for all other sessions will be the same. Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be the set of parties participating in the execution of Π corresponding to the session identified by the session identifier sid . Also let $\mathcal{P}^{\mathcal{A}} \subseteq \mathcal{P}$ be the set of parties corrupted by the adversary \mathcal{A} at any time. Recall that we are in the setting of adaptive corruption so more parties could be added to this set as the protocol proceeds. At any point \mathcal{S} only generates messages on behalf

of parties $\mathcal{P} \setminus \mathcal{P}^{\mathcal{A}}$. In the following, if at the end of some round all parties are corrupted then \mathcal{S} does not need to go to do anything else.

Round 1 Messages $\mathcal{S} \rightarrow \mathcal{A}$: In the first round \mathcal{S} must generate messages on behalf of the honest parties, i.e. parties in the set $\mathcal{P} \setminus \mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P} \setminus \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as:

1. Generate a fake commitment $(\beta_i, state_i) \leftarrow \text{Equiv}_1(CK, \nu)$.

It then sends β_i to \mathcal{A} on behalf of party P_i and it internally saves $state_i$.

Round 1 Messages $\mathcal{A} \rightarrow \mathcal{S}$: Also in the first round the adversary \mathcal{A} generates the messages on behalf of corrupted parties in $\mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as follows:

1. Extracting inputs of corrupted parties: Let β_i be the commitment that \mathcal{A} sends on behalf of P_i . Our simulator \mathcal{S} runs the extraction algorithm $\text{Extr}(CK, \mu, \beta_i)$ in order to obtain x_i .
Note that it is possible that \mathcal{A} sends a commitment β_i on behalf of P_i such that it is not well-formed, or in other words extraction using the function Extr fails. In this case \mathcal{S} sets $x_i = \perp$ and proceeds to the next step. (Looking ahead, we note that in this case the adversary will not be able to generate a valid second round message.)
2. Next \mathcal{S} sends $(\text{input}, \text{sid}, \mathcal{P}, P_i, x_i)$ to \mathcal{F}_f on behalf of the corrupted party P_i .

Simulating corruption of parties in Round 1: When \mathcal{A} corrupts a real world party P_i , then \mathcal{S} first corrupts the corresponding ideal world party P_i and obtains its input x_i . Next \mathcal{S} prepares the internal state on behalf of P_i such that it will be consistent with the commitment value β_i that it had provided to \mathcal{A} earlier. Specifically \mathcal{S} computes $\text{Equiv}_2(CK, \nu, \beta_i, state_i, x_i)$ in order to obtain randomness ω_i such that $\beta_i = \text{Com}(CK, \beta_i; \omega_i)$. \mathcal{S} provides ω_i as the randomness of party P_i to \mathcal{A} . Note that \mathcal{S} can do this at any point during 1st round.

Completion of Round 1: After \mathcal{S} has submitted the inputs of all the corrupted parties to \mathcal{F}_f then it responds by sending back the message $(\text{output}, \text{sid}, \mathcal{P}, out)$ where $out = f(\{x_j\}_{j \in [n]})$. Note that in case \mathcal{S} had failed to extract an input for some player P_i then it would have sent \perp to \mathcal{F}_f and would have received \perp as the output from the ideal functionality.

Round 2 Messages $\mathcal{S} \rightarrow \mathcal{A}$: In the second round \mathcal{S} generates messages on behalf of the honest parties, i.e. parties in the set $\mathcal{P} \setminus \mathcal{P}^{\mathcal{A}}$ as follows:

1. For each party $P_i \in \mathcal{P} \setminus \mathcal{P}^{\mathcal{A}}$, \mathcal{S} generates $c_i = \text{Enc}(pk_0, \phi^{\ell_{in}} || out)$, $c'_i = \text{Enc}(pk_1, \phi^{\ell_{in}} || out)$ and generates π_i as a simulated proof for the statement $(c_i, c'_i, (i, \{\beta_j\}_{j \in [n]}))$. More specifically it generates $\pi_i \leftarrow S_2(\sigma, \tau, (c_i, c'_i, (i, \{\beta_j\}_{j \in [n]})))$.

\mathcal{S} sends (c_i, c'_i, π_i) to \mathcal{A} on behalf of P_i .

Round 2 Messages $\mathcal{A} \rightarrow \mathcal{S}$: In the second round the adversary \mathcal{A} generates the messages on behalf of corrupted parties in $\mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as:

1. Let (c_i, c'_i, π_i) be the message that \mathcal{A} sends on behalf of party P_i . \mathcal{S} checks to see if $V(\sigma, (c_i, c'_i, (i, \{\beta_j\}_{j \in [n]})), \pi_i) = 1$ for each $P_i \in \mathcal{P}^{\mathcal{A}}$.

If all the proofs verify then \mathcal{S} sends the message $(\text{generateOutput}, \text{sid}, \mathcal{P})$ to the ideal functionality \mathcal{F}_f .

Simulating corruption of parties during/at the end of Round 2: When \mathcal{A} corrupts a party P_i in the real world, then \mathcal{S} corrupts the corresponding party P_i in the ideal world and obtains its input x_i . Next \mathcal{S} prepares the internal state on behalf of P_i such that it will be consistent with messages it had sent on behalf of P_i . As explained before, \mathcal{S} generates randomness ω_i that explains the commitment β_i to the value x_i running the algorithm $\omega_i = \text{Equiv}_2(\text{CK}, \nu, \beta_i, \text{state}_i, x_i)$. Next \mathcal{S} needs to explain the second round message (c_i, c'_i, π_i) . \mathcal{S} has to explain the message (c_i, c'_i, π_i) by computing the randomness as $\psi_i = \text{Explain}_{\text{DDK}}(\text{PK}, (c_i, c'_i, \pi_i), (x_i || \phi^{\text{out}}, x_i || \phi^{\text{out}}, (i, \{\beta_j\}_{j \in [n]}), (0^{n \cdot \ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j \in [n]}); u)$ where $t_i = \omega_i$ and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. \mathcal{S} provides $\omega_i || \psi_i$ as the randomness of party P_i to \mathcal{A} . Note that \mathcal{S} can do this at any point during or after the round 2 of the protocol.

This completes the description of the simulator.

5 Proof of Security

In this section, via a sequence of hybrids, we will prove that no environment \mathcal{Z} can distinguish the ideal world experiment with \mathcal{S} and \mathcal{F}_f (as defined above) from a real execution of Π with \mathcal{A} . We will start with the real world execution in which the adversary \mathcal{A} interacts directly with the honest parties holding their inputs and step-by-step make changes till we finally reach the simulator as described in Section 4. At each step we will argue that the environment cannot distinguish the change except with negligible probability.

Hybrid 0. This hybrid corresponds to the \mathcal{Z} interacting with the real world adversary \mathcal{A} and honest parties that hold their private inputs.

We can restate the above experiment with the simulator as follows. We replace the real world adversary \mathcal{A} with the ideal world adversary \mathcal{S} . The ideal adversary \mathcal{S} starts by invoking a copy of \mathcal{A} and running a simulated interaction of \mathcal{A} with the environment \mathcal{Z} and the honest parties. \mathcal{S} forwards the messages that \mathcal{A} generates for its environment directly to \mathcal{Z} and vice versa (as explained in the description of the simulator \mathcal{S}). In this hybrid the simulator \mathcal{S} holds the private inputs of the honest parties and generates messages on their behalf using the honest party strategies as specified by Π .

Hybrid 1. In this hybrid, we change how the internal randomness of the corrupted party is explained to \mathcal{A} on being adaptively corrupted. Specifically we change the randomness that is used to explain the ciphertext \mathcal{S} generates on behalf of parties in round 2 of protocol Π .

Recall that in the second round \mathcal{S} on behalf of an honest party P_i generates the second message as $(c_i, c'_i, \pi_i) = \text{DenEnc}_{\text{DDK}}(PK, x_i || \phi^{\ell_{out}}, x_i || \phi^{\ell_{out}}, (i, \{\beta_j\}_{j \in [n]}), (0^{n \cdot \ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j \in [n]}); r_i)$ where t_i is the randomness used in generating commitment β_i and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. So if \mathcal{A} corrupts P_i then the randomness r_i would be reveal to \mathcal{A} . In Hybrid 1, instead we provide $\psi_i = \text{Explain}_{\text{DDK}}(PK, (c_i, c'_i, \pi_i), (x_i || \phi^{\ell_{out}}, x_i || \phi^{\ell_{out}}, (i, \{\beta_j\}_{j \in [n]}), (0^{n \cdot \ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j \in [n]}); u)$ where t_j values are as before.

Lemma 1. $\text{Hybrid}_0 \approx_c \text{Hybrid}_1$.

Proof. The indistinguishability of Hybrid_1 from Hybrid_0 follows from the indistinguishability of explanation property of the Double Key deniable encryption scheme.

Hybrid 2. In this hybrid we change the way \mathcal{S} generates the message (c_i, c'_i, π) on behalf of the honest parties.

Recall that in the second round in Hybrid 1, \mathcal{S} on behalf of an honest party P_i generates the second message as $(c_i, c'_i, \pi_i) = \text{DenEnc}_{\text{DDK}}(PK, x_i || \phi^{\ell_{out}}, x_i || \phi^{\ell_{out}}, (i, \{\beta_j\}_{j \in [n]}), (0^{n \cdot \ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j \in [n]}); r_i)$ where t_i is the randomness used in generating commitment β_i and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. We will change this by generating the ciphertexts directly using procedures Enc and the prover P .

Specifically, $c_i = \text{Enc}(pk_0, x_i || \phi^{\ell_{out}}; s_0^i)$ and $c'_i = \text{Enc}(pk_1, x_i || \phi^{\ell_{out}}; s_1^i)$ and outputs (c_i, c'_i, π_i) , where $\pi_i \leftarrow P(\sigma, (c_i, c'_i, \{i, \{\beta\}_{j \in [n]}\}), (x_i || \phi^{\ell_{out}}, x_i || \phi^{\ell_{out}}, s_0^i, s_1^i, (0^{n \cdot \ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j \in [n]})))$ where t_i is the randomness used in generating commitment β_i and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$.

Lemma 2. $\text{Hybrid}_1 \approx_c \text{Hybrid}_2$.

Proof. The indistinguishability of Hybrid_2 from Hybrid_1 follows immediately from the indistinguishability of *source of ciphertext* property of the Double Key deniable encryption scheme.

Hybrid 3. In this hybrid, we change how σ , which is a part of PK , and the proofs π_i for every $P_i \in \mathcal{P} \setminus \mathcal{P}^{\mathcal{A}}$ are generated.

More specifically, \mathcal{S} runs the setup algorithm $\text{Setup}_{\text{DDK}}(1^\lambda, 1^{\ell_{in} + \ell_{out}})$ of the Double Key deniable encryption scheme, but replaces its internal call to the algorithm K with $S = (S_1, S_2)$ of the NIZK proof system. More specifically, \mathcal{S} generates $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \text{Setup}(1^\lambda)$, $(\sigma, \tau) \leftarrow S_1(1^\lambda)$, along with the public denying key DK . It sets the public key $PK = (\sigma, pk_0, pk_1, DK)$. We also generate fake proofs π_i using trapdoor τ . Specifically we generate $\pi_i \leftarrow S_2(\sigma, \tau, (c_i, c'_i, (i, \{\beta_j\}_{j \in [n]})))$.

Lemma 3. $\text{Hybrid}_2 \approx_c \text{Hybrid}_3$.

Proof. The indistinguishability of Hybrid₃ from Hybrid₂ follows immediately from the computational zero-knowledge property of the NIZK proof system.

Hybrid 4. We don't change anything in the output of the hybrid itself. We just use knowledge of μ to extract the inputs \mathcal{A} commits to in the 1st round messages that it sends on behalf of the corrupted parties. More specifically, \mathcal{S} for every $P_i \in \mathcal{P}^A$ obtains $x_i = \text{Extr}(CK, \mu, \beta_i)$. If extraction fails then it sets $x_i = \perp$.

Hybrid 5. In this hybrid, we change how the simulator \mathcal{S} generates c'_i in the second round message (c_i, c'_i, π_i) on behalf of honest parties $P_i \in \mathcal{P} \setminus \mathcal{P}^A$. In particular, \mathcal{S} instead of computing the ciphertext $c'_i = \text{Enc}(pk_1, x_i || \phi^{\ell_{out}}; s_1^i)$, generates $c'_i = \text{Enc}(pk_1, \phi^{\ell_{in}} || out; s_1^i)$, where out is the output computed as $f(\{x_j\}_{j \in [n]})$ using the inputs x_i of the honest parties, that the simulator has access to, and extracted inputs of the malicious parties.

Lemma 4. Hybrid₄ \approx_c Hybrid₅.

Proof. We base the indistinguishability between hybrids Hybrid₄ and Hybrid₅ on the semantic security of the encryption scheme (Setup, Enc, Dec).

Hybrid 6. In this hybrid we essentially reverse the change that was made in going from Hybrid 2 to Hybrid 3. In particular we change the σ so that it is sampled from the honest distribution and generate the proof honestly. Note that since now we have changed the ciphertext c'_i the proof will have to be generated with respect to language \mathcal{L}_2 .

More specifically, \mathcal{S} uses K to generate σ instead of S_1 . Also for every $P_i \in \mathcal{P} \setminus \mathcal{P}^A$, \mathcal{S} generates $\pi_i \leftarrow P(\sigma, (c_i, c'_i, (i, \{\beta_j\}_{j \in [n]})), (x_i || \phi^{\ell_{out}}, \phi^{\ell_{in}} || out, s_0^i, s_1^i, (\{x_i\}_{i \in [n]}, out, \{t_j\}_{j \in [n]})))$ where t_j is the witness that $\beta_j \in \mathcal{L}_{\text{Com}}$.

Lemma 5. Hybrid₅ \approx_c Hybrid₆.

Proof. The indistinguishability of Hybrid₅ from Hybrid₆ follows immediately from the computational zero-knowledge property of the NIZK proof system.

Hybrid 7. In this hybrid we change how $\mathfrak{o}P$, the obfuscated program in the CRS is generated. More specifically, $\mathfrak{o}P$ is generated as an obfuscation of $\text{Prog}_{sk_1, PK, CK, f}$ instead of $\text{Prog}_{sk_0, PK, CK, f}$.

In the following we show that the program Prog is equivalent under sk_0 and sk_1 with overwhelming probability. This allows us to conclude that the Hybrid 6 and Hybrid 7 are indistinguishable based on indistinguishability obfuscation.

Lemma 6. $\text{Prog}_{sk_0, PK, CK, f} \equiv \text{Prog}_{sk_1, PK, CK, f}$.

Proof. Recall that the underlying language \mathcal{L} of the Double Key deniable encryption scheme consists of two languages, namely \mathcal{L}_1 and \mathcal{L}_2 . Note that since the NIZK has statistical soundness with overwhelming probability over the choices of σ we have that all ciphertexts with an accepting proof must

be from one of the two languages. We refer to the two types of ciphertexts corresponding to the language \mathcal{L}_1 and \mathcal{L}_2 , as Type-1 and Type-2 ciphertext, respectively.

Recall that the program **Prog** takes $\{\beta_j\}_{j \in [n]}$ and $\{c_j, c'_j, \pi_j\}_{i \in [n]}$ as input. Recall from Figure 2 that in Step 1, **Prog** checks to see that all the proofs π_i are accepting and otherwise it outputs \perp . This means that for the program to do anything interesting all the proofs must be valid. Next we will show that in such cases the output of the program is identical regardless of whether sk_0 or sk_1 is used.

All ciphertexts are of Type-1: In this case, c_j and c'_j for $j \in [n]$ encrypted under pk_0 and pk_1 respectively, encrypt the same value. Hence, regardless of whether sk_0 is used or sk_1 is used the program outputs the exact same value $f(\{x_j\}_{j \in [n]})$.

There is at least one Type-2 ciphertext: Note that, in case sk_0 is used then we have that Step 2 of **Prog** is never invoked. On the other hand in case sk_1 is used then we have that Step 2 of **Prog** is necessarily invoked.

In other words if sk_0 is used then the x_j values are decrypted and output is calculated. On the other hand if sk_1 is used then a hard-coded *out* value is generated. We will argue that in both cases the output generated by **Prog** is identical. We argue this by showing that the only acceptable value for the hard-coded value *out* is $f(\{x_j\}_{j \in [n]})$, where x_j are the inputs parties commit to in the first round. Recall that the commitment scheme is perfectly binding, meaning that for every commitment β_i there is exactly one x_i such that $(\beta_i, x_i) \in \mathcal{L}_{\text{COM}}$. This proves our claim. \square

Hybrid 8. In this hybrid we do the same change that was made in going from Hybrid 2 to Hybrid 3. In this hybrid, we change how σ , which is a part of PK , and the proofs π_i for every $P_i \in \mathcal{P} \setminus \mathcal{P}^A$ are generated.

More specifically, \mathcal{S} runs the setup algorithm $\text{Setup}_{\text{DDK}}(1^\lambda, 1^{\ell_{in} + \ell_{out}})$ of the Double Key deniable encryption scheme, but replaces its internal call to the algorithm K with $S = (S_1, S_2)$ of the NIZK proof system. More specifically, \mathcal{S} generates $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \text{Setup}(1^\lambda)$, $(\sigma, \tau) \leftarrow S_1(1^\lambda)$, along with the public denying key DK . It sets the public key $PK = (\sigma, pk_0, pk_1, DK)$. We also generate fake proofs π_i using trapdoor τ . Specifically, it generates $\pi_i \leftarrow S_2(\sigma, \tau, (c_i, c'_i, (i, \{\beta_j\}_{j \in [n]})))$.

Lemma 7. $\text{Hybrid}_7 \approx_c \text{Hybrid}_8$.

Proof. The indistinguishability of Hybrid_7 from Hybrid_8 follows immediately from the computational zero-knowledge of the NIZK proof system.

Hybrid 9. In this hybrid, we change how the simulator \mathcal{S} generates c_j in the second round message (c_j, c'_j, π_j) on behalf of honest parties $P_j \in \mathcal{P} \setminus \mathcal{P}^A$. More specifically, \mathcal{S} instead of computing $c_j = \text{Enc}(pk_0, x_i | \phi^{\ell_{out}})$, it computes $c_j = \text{Enc}(pk_0, \phi^{\ell_{in}} | \text{out})$ where $\text{out} = f(\{x_j\}_{j \in [n]})$.

Lemma 8. $\text{Hybrid}_8 \approx_c \text{Hybrid}_9$.

Proof. We base the indistinguishability between hybrids Hybrid_8 and Hybrid_9 on the semantic security of the encryption scheme, ($\text{Setup}, \text{Enc}, \text{Dec}$).

Hybrid 10. In this hybrid we change the way the public parameters of the commitment scheme COM are generated. In particular, \mathcal{S} runs the setup algorithm $\text{Setup}_{\text{Com}}^{\text{equiv}}(1^\lambda)$ (instead of $\text{Setup}_{\text{Com}}^{\text{bind}}(1^\lambda)$) of the adaptively secure commitment scheme COM and obtains (CK, μ, ν) where the trapdoor μ is still being used for extraction of adversary’s inputs.

Lemma 9. $\text{Hybrid}_9 \approx_c \text{Hybrid}_{10}$.

Proof. Indistinguishability between hybrids Hybrid_9 and Hybrid_{10} follows from the indistinguishability of the public parameters of the commitment scheme COM.

Hybrid 11. In this hybrid we change the way \mathcal{S} generates the commitments on behalf of the honest parties. In particular we will remove the inputs and make these commitments equivocal. More specifically, for every party $P_i \in \mathcal{P} \setminus \mathcal{P}^A$ the first round message is computed by \mathcal{S} running $(\beta_i, \text{state}_i) \leftarrow \text{Equiv}_1(CK, \nu)$. If the party later gets corrupted then \mathcal{S} will produce randomness ω_i to equivocate the commitment β_i to the prescribed input x_i . To this end, \mathcal{S} will run $\omega_i = \text{Equiv}_2(CK, \nu, \beta_i, \text{state}_i, x_i)$.

Lemma 10. $\text{Hybrid}_{10} \approx_c \text{Hybrid}_{11}$.

Proof. We base the indistinguishability between hybrids Hybrid_{10} and Hybrid_{11} on the polynomial equivocality of the adaptively secure commitment scheme COM.

Note that Hybrid_{11} is identical to the simulation strategy described in Section 4. This concludes the proof.

6 Extending to Leakage Tolerant Secure Computation

The adaptively secure protocol presented in this paper also turns out to be leakage tolerant. The model of leakage can be found in the full version [26].

Lemma 11. *Assume the existence of indistinguishability obfuscation and doubly enhanced trapdoor permutation then any ideal functionality \mathcal{F}_f can be UC-securely realized in the \mathcal{F}_{CRS} -model against any adaptive, active adversary corrupting an arbitrary number of parties and allowed with arbitrary leakage. Furthermore this protocol involves only two rounds of broadcast.*

This lemma follows immediately from our construction and proof except for some syntactic differences. We explain this next. We will only describe how our simulator for adaptive security (from Section 4) can be converted into a

simulator for the setting of leakage tolerance. The proof of indistinguishability for the adaptive simulator was already provided in Section 5.

Recall that the simulator for arguing adaptive security, on corruption of an honest party, uses the honest party's input alone in order to explain the messages it had previously sent on behalf of the honest party. In the setting of leakage, we note that this method of explanation can directly be expressed by a circuit that on input the input of the honest party outputs the internal secret state of that party. Furthermore note that the way in which the simulator explains its first round messages of honest parties remains the same even after it has sent the second round messages.

Using this explanation procedure as a translation method, allows us to immediately conclude that any leakage query of the real-world adversary can be reduced directly to a leakage query in the ideal-world.

Acknowledgments

We would like to thank Oxana Poburinnaya and Ran Canetti for pointing out that sub-exponential iO is needed to instantiate our double key deniable encryption scheme.

References

1. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.
2. Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. *Cryptology ePrint Archive*, Report 2011/613, 2011. <http://eprint.iacr.org/2011/613>.
3. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.
4. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.
5. Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 266–284, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Berlin, Germany.
6. Manuel Blum, Paul Feldman, and Silvio Micali. Proving security against chosen cyphertext attacks. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 256–268, Santa Barbara, CA, USA, August 21–25, 1988. Springer, Berlin, Germany.

7. Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 316–334, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Germany.
8. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press.
9. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadelphia, Pennsylvania, USA, May 22–24, 1996. ACM Press.
10. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany.
11. Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2014:845, 2014.
12. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany.
13. Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st Annual Symposium on Foundations of Computer Science*, pages 541–550, Las Vegas, Nevada, USA, October 23–26, 2010. IEEE Computer Society Press.
14. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.
15. Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. Adaptively secure, universally composable, multi-party computation in constant rounds. *IACR Cryptology ePrint Archive*, 2014:858, 2014.
16. Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkatasubramanian. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 316–336, Bangalore, India, December 1–5, 2013. Springer, Berlin, Germany.
17. Ivan Damgård, Carmit Hazay, and Arpita Patra. Leakage resilient secure two-party computation. *IACR Cryptology ePrint Archive*, 2011:256, 2011.
18. Ivan Damgård, Antigoni Polychroniadou, and Vanishree Rao. Adaptively secure UC constant round multi-party computation protocols. *IACR Cryptology ePrint Archive*, 2014:830, 2014.
19. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317, St. Louis, Missouri, October 22–24, 1990. IEEE Computer Society Press.

20. Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO’89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544, Santa Barbara, CA, USA, August 20–24, 1989. Springer, Berlin, Germany.
21. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17, Athens, Greece, May 26–30, 2013. Springer, Berlin, Germany.
22. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany.
23. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.
24. Sanjam Garg, Divya Gupta, Dakshita Khurana, and Amit Sahai. All-but-one leakage resilient multiparty computation and incoercible multiparty computation. Personal Communication, 2014.
25. Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 297–315, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany.
26. Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2014:844, 2014.
27. Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 105–123, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.
28. Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
29. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
30. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, New York, USA, May 25–27, 1987. ACM Press.
31. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany.
32. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
33. Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *Advances*

- in *Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 132–150, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany.
34. Carmit Hazay and Arpita Patra. One-sided adaptively secure two-party computation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 368–393, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany.
 35. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.
 36. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 335–354, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.
 37. Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *35th Annual ACM Symposium on Theory of Computing*, pages 683–692, San Diego, California, USA, June 9–11, 2003. ACM Press.
 38. Yehuda Lindell and Benny Pinkas. A proof of security of Yao’s protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.
 39. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.
 40. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.
 41. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.