Towards Accountability in CRS Generation*

Prabhanjan Ananth† Gilad Asharov‡ Hila Dahari§ Vipul Goyal ¶

August 24, 2021

Abstract

It is well known that several cryptographic primitives cannot be achieved without a common reference string (CRS). Those include, for instance, non-interactive zero-knowledge for NP, or maliciously secure computation in fewer than four rounds. The security of those primitives heavily relies upon on the assumption that the trusted authority, who generates the CRS, does not misuse the randomness used in the CRS generation. However, we argue that there is no such thing as an unconditionally trusted authority and every authority must be held accountable for any trust to be well-founded. Indeed, a malicious authority can, for instance, recover private inputs of honest parties given transcripts of the protocols executed with respect to the CRS it has generated.

While eliminating trust in the trusted authority may not be entirely feasible, can we at least move towards achieving some notion of accountability? We propose a new notion in which, if the CRS authority releases the private inputs of protocol executions to others, we can then provide a publicly-verifiable proof that certifies that the authority misbehaved. We study the feasibility of this notion in the context of non-interactive zero knowledge and two-round secure two-party computation.

*A preliminary version of this work appears in EUROCRYPT 2021.

†University of California, Santa Barbara, prabhanjan@cs.ucsb.edu.

‡Bar-Ilan University, Israel. Gilad.Asharov@biu.ac.il.

§Weizmann Institute, Israel. hila.dahari@weizmann.ac.il.

¶NTT Research and Carnegie Mellon University. vipul@cmu.edu.

Contents

1 Introduction

Very broadly, cryptography can be seen as having two parallel lines of research: one where the parties don't trust anyone but themselves, and another where security relies on some kind of trust assumption. Most notably, many works have relied on the common reference string (CRS) model where a trusted party chooses and publishes a public string. The advantage of relying on a CRS depends upon the setting. For example, for ZK it is known that while in the CRS model, a non-interaction solution can be achieved [BFM88, FLS99] one needs at least 3 rounds in the plain model [GO94]. For MPC, two rounds are sufficient in the CRS model [MW16, GS18, BL18] while the best known constructions in the plain model require at least 4-rounds [KO04, ACJ17, BHP17, BGJ+18, HHPV18, CCG+20]. Furthermore, UC security is known to be impossible to achieve in the plain model [CF01, CKL03] while this impossibility can be bypassed in the CRS model [CF01, CLOS02]. Thus, while one might prefer to obtain constructions in the plain model, it seems unlikely that the CRS model will be abandoned anytime soon.

Do There Really Exist Trusted Parties? We argue that in real life, there is no such thing as an unconditional trusted party. We argue that the only reason we trust a party is because the cost of cheating (if caught) for that party would be much higher than the potential gains obtained by cheating successfully. Indeed this applies everywhere in our society. We are more comfortable trusting a large bank with our personal information compared to a small individual lender only because a large bank will pay a much higher cost (loss of reputation and potential future business) if it behaves maliciously. However if the cost to the large bank was zero, the reasons for placing this trust would be unfounded. There are multiple examples where even large entities systematically participated in an activity which would be generally unacceptable only because they thought the activity would not become public knowledge (e.g., Facebook selling data to Cambridge Analytica, or Wells Fargo opening accounts without customer knowledge).

Compared to real life, in cryptography, life is largely black and white: dishonest parties can be arbitrarily bad and trusted parties are unconditionally trusted. For example, the party generating a CRS (referred to as the CRS authority from hereon) in a NIZK system can potentially even recover your witness entirely and sell it for profit. Similarly, in MPC, the CRS authority may recover your input and pass it on to another party. Even if you detect that the authority is doing that, it's not clear how to prove it in a court of law and seek damages. You can publicly blame the authority for doing that. But this is then indistinguishable from a malicious party blaming an honest authority.

These concerns have motivated the study of weaker notions such as ZAPs [DN00] and super-polynomial simulation security [Pas03]. Groth and Ostrovsky studied the so called multi-string model [GO07] where multiple authorities publish common reference strings such that a majority of them are guaranteed to be honest. Goyal and Katz [GK08], and later Garg et. al [GGJS11] studied UC security with an unreliable CRS if the CRS turns out to be malicious, some other setup or an honest majority can come to the rescue. Bellare et. al [BFS16] studied NIZKs with an untrusted CRS where even if the CRS is malicious, some weaker security properties still hold.

In this work, we focus on a single CRS while providing some notion of accountability towards the CRS generation authority. Our direction is orthogonal to many of the works mentioned and, to our knowledge, largely unexplored.

Towards Accountability in CRS Generation: While eliminating trust in the CRS authority entirely may not be feasible, can we at least move towards achieving some notion of accountability? As an example, suppose you find out that the CRS authority decrypted your input used in an execution of MPC protocol and sold to another party for profit. Can you obtain a cryptographic proof of this fact? Can you convince others that such an incident has happened? Indeed there are

limits on what can and cannot be achieved. For example, if the authority sells your input and you never find out, it's unclear if something can be done. But if the decrypted input indeed falls into your hand (e.g., the person buying the input from the authority was your own agent), you know for sure that the authority is dishonest (although you may not be able to prove it to others).

In this work, we study if there is *any* meaningful notion of accountability that can be achieved with respect to a CRS authority. We focus specifically on the case of NIZK, and two-round MPC which are both known to be impossible in the plain model and yet achievable with a CRS. Our work runs into several novel technical challenges as we discuss later. We note that our study is far from complete and leaves open various intriguing questions.

1.1 Our Results

In this work, we propose novel notions of accountability in the context of two party secure computation protocols and NIZKs. We also accompany these definitions with constructions realizing these notions.

Secure Two-Party Computation (2PC). Our definition of malicious authority security first requires the same security guarantees as in regular secure computation, that is, if the CRS was honestly generated, then the protocol achieves simulation security in the presence of a malicious adversary. To capture the setting when the CRS authority is malicious, we require the following two security properties:

- **Accountability.** Suppose the authority generated a CRS maliciously. At a later point in time, it offers a service to recover the honest parties' inputs from the transcripts of protocol executions between these parties, using the trapdoors it embedded in the CRS. The accountability property guarantees that we can hold such a CRS authority accountable by producing a piece of publicly verifiable evidence that incriminates this authority for its malpractice. This evidence can then be presented in a court of law to penalize this authority. We formalize this by defining an efficient extractor that can interact[1] with this malicious authority and outputs a piece of evidence (a string). We associate with the scheme an algorithm Judge, which then determines whether this evidence is valid or not.
  The authority should not distinguish whether it interacts with the extractor (who is trying to incriminate the authority) or with a real party (who is trying to learn the inputs of the honest parties). Note that if the authority has some auxiliary information about the honest party's input, it can possibly produce the input without using the CRS trapdoor at all. In that case, it seems impossible to obtain incriminating evidence from the response of the authority. To avoid this issue, we specify a distribution $\mathcal{D}$ such that the inputs of the honest parties are sampled from this distribution in the security experiment. We stress that this requirement is only for the accountability security experiment. Our construction satisfies the usual definition of 2PC (without requiring any distribution on the inputs) in case the CRS is honest.
- **Defamation-free.** Of course, accountability cannot stand by itself. This notion opens up the possibility of falsely accusing even an honest CRS authority (who never partakes in running the input recovery service mentioned above) of malpractice. We complement the definition of accountability by defining another property called *defamation-free*. Roughly speaking, this definition states that just given an honestly generated CRS, it should be computationally infeasible to come up with an evidence that would incriminate an honest authority.

[1] We stress that this extractor interacts with the malicious authority online *without being able to rewind the authority*. This is because, if we want to implicate the authority in the real world then we would not have the ability to rewind such an authority.

We study two variants of accountability. First, we study the scenario mentioned above in which two parties engage in a secure protocol. Then, one of them comes to the authority *after the fact* and asks to open the honest party's input. That party has to provide to the authority its view, which includes its own input and randomness. In the second (stronger) definition, we imagine the authority will be more cautious and refuse to answer such queries. Instead, the authority will insist on being involved from the beginning. In this model, the authority completely controls one of the parties, actively participates in the protocol execution on behalf of this party, and finally recovers and provides to this party the honest party's input. We refer to these notions as weak and strong accountability.

Impossibility Result. The first question is whether this new notion can be realized at all. Unfortunately, we show that even the weak definition cannot be realized for all functions. We show the following.

Theorem 1.1 (Informal)**.** *There exists a two-party functionality $\mathcal{F}$ such that there does not exist any secure two-party computation protocol for $\mathcal{F}$ in the CRS model satisfying both (weak) accountability and defamation-free properties.*

Specifically, the class of functionalities for which the above impossibility result hold are functionalities where given the output, we can efficiently recover the inputs. Indeed, an impossibility result is easy to see in this case since the authority can recover the input without even using any trapdoor related to the CRS (and in fact, anyone can recover the input of any party). Since this class of functionalities is somewhat trivial, and such functions are usually considered as functions where secure computation is not necessary (a trivial protocol where a party just gives its input suffices), this gives us hope that we can come up with positive results for large class of interesting functionalities. We focus on the setting of maliciously secure *two-round* two-party since that is known to be impossible to achieve in the plain model.

Construction. We then study the following class of (asymmetric) two-party functionalities $\mathcal{F}$: the two-party functionality takes as input (x, y) and outputs $g(\{x_i\}_{y_i=1})$ to the second party (with input y) for some function g. That is, it outputs g on only those bits of x that are indexed by the bits of y set to 1. This class of functions includes for instance, oblivious transfer, private information retrieval, subset sum, and more. We show the following:

Theorem 1.2 (Informal)**.** *Assuming SXDH (Symmetric External Diffie-Hellman) on bilinear maps, there exists a two-round maliciously secure two-party computation protocol for $\mathcal{F}$ satisfying both weak accountability (with respect to the uniform distribution over the inputs) and defamation-free.*

Indeed, obtaining such a construction turns out to be surprisingly non-trivial and requires one to overcome novel technical challenges. We refer the reader to Section 2 (Technical Overview) for a summary of techniques.

Strong accountability for Oblivious Transfer. As mentioned, we study weak and strong accountability, depending on whether the malicious authority actively participates in the protocol execution or not. We focus on the oblivious transfer functionality and demonstrate that strong accountability is possible to achieve, based on a (seemingly) stronger assumption.

Theorem 1.3 (Informal)**.** *Assuming indistinguishability obfuscation for P/poly [BGI+01, GGH+16] and SXDH in bilinear groups, there exists a two-round maliciously secure oblivious transfer protocol in the CRS model satisfying both strong accountability and defamation-free properties, with respect to the uniform distribution over the inputs.*

The techniques developed in the above construction can potentially be extended also for the class of functions in $\mathcal{F}$ for which Theorem 1.2 holds, although we focused on oblivious transfer for simplicity.

Non-Interactive Zero-Knowledge (NIZK). Another basic cryptographic primitive which relies on a CRS is NIZK. Indeed, CRS shows up in several cryptographic constructions primarily because they use NIZK as a building block. Similar to the 2PC case, we require the same guarantees as a regular NIZK when the CRS is honestly generated, namely, completeness, soundness and zero-knowledge. We associate with the proof system a Judge algorithm and require accountability and defamation free properties:

- **Accountability.** For any CRS^* that might be maliciously produced by the CRS authority, if there exists an adversary that upon receiving pairs (x, π) can recover the witness w, (where x is an instance, π is a proof that x is in the associated language, and w is the secret witness), then there exists an extractor that can create a piece of evidence τ that is accepted by Judge. As before, our accountability property is parameterized by a distribution $\mathcal{D}$ defined on the instance-witness pairs. Indeed this is necessary since if the authority can guess the witness without using the CRS trapdoor, the security guarantees we have in mind are impossible to achieve.
- **Defamation-free.** This states that no non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ upon receiving a CRS that was honestly generated can come up with a piece of evidence τ that makes Judge accept.

We consider an NP language L consisting of instances of the form $(C, c_1, \ldots, c_n, b)$ such that $C : \{0,1\}^n \to \{0,1\}$ is a boolean circuit, each c_i is a commitment of x_i and moreover, $C(x_1, \ldots, x_n) = b$. We note that we can reduce any NP-complete language to this language based on the existence of rerandomizable commitments. We show the following.

Theorem 1.4 (Informal)**.** *Assuming SXDH on bilinear maps, there exists a NIZK for L in the CRS model satisfying both the accountability and the defamation-free properties.*

We can handle a class of distributions $\mathcal{D}$ with the only requirement being that a distribution in this class computes the commitments using uniform randomness while the circuit C and inputs $(x_1, \ldots, x_n)$ can be arbitrarily chosen.

Open Problems. We believe that a systematic study of the notions of accountability in the CRS model is an exciting line of research. Our work leaves open several natural questions. Can we broaden and characterize the class of functionalities for which accountable 2PC can be achieved? Can we extend our construction to more than two parties? While we focus on two rounds, obtaining even three round constructions would be valuable since the best known constructions in the plain model require at least four rounds.

One could also consider stronger notions where the authority only supplies some information about the input (e.g. the first bit of the input) rather than the entire input. In this setting, it seems the extractor would need to obtain multiple responses from the authority and somehow combine them into a single proof. Furthermore, while we focus on privacy (of the input in case of secure computation, or the witness in case of NIZK) in this work, what if the authority instead attacks correctness or soundness?

Another interesting direction is to consider other settings where CRS is used such as obtaining UC security.

Related Works. Our notion is inspired, in part, by broadcast encryption with traitor tracing [CFN94] where, given a decryption box, there is a trace algorithm (similar to our Judge algorithm)

which identifies the cheating party. However there are crucial differences. Our Judge algorithm does not have direct access to the CRS authority and only gets to see a string produced by the extractor. Furthermore, our extractor only gets to interact with the CRS authority *online* and, in particular, *does not get to rewind the CRS authority*. In another related line of research, Goyal [Goy07] introduced what is known as accountable authority (AA) identity-based encryption (IBE) where if the authority generating the IBE public parameters is dishonest and releases a "decryption box", the authority can be implicated in a court. Our definition is also inspired by public verifiability in covert security, introduced by Asharov and Orlandi [AO12]. This definition shows how one can extract, given a transcript of the protocol, a piece of evidence showing that there was a misbehavior in the execution. The definition also requires defamation free, so that innocents cannot be implicated.

Another related notion of our work is subversion security, suggested by Bellare, Fuchsbauer, and Scafuro [BFS16] (see also [Fuc18]). The work studies the security of NIZKs in the presence of a maliciously chosen common reference string. It shows that several security properties can still be preserved when the CRS is maliciously generated. Nevertheless, it is shown that zero-knowledge cannot be preserved simultaneously with soundness when the common reference string is maliciously generated. Our work takes a different approach and seeks accountability when such misbehavior is detected. It will be intriguing to see how these two notions can intertwine.

2 Technical Overview

We start by explaining the main idea that underpins the constructions of NIZK, oblivious transfer and secure two-party computation with malicious authority security. Realizing this insight will lead us to different challenges in the context of designing each of the different primitives; we discuss the challenges for each of these primitives separately.

Main Idea. Our main idea is to force the CRS authority to include a transcript of execution of the protocol as part of the CRS, where the transcript has a secret s embedded inside. In the context of NIZKs, we force the authority to include a NIZK proof in the CRS where the witness contains the secret s. In the case of oblivious transfer and secure two-party computation, the sender and the receiver's input in the transcript are generated as a function of s.

In the honest execution, the transcript in the CRS is ignored. However, to argue accountability, the extractor will cleverly maul this transcript in the CRS to generate another transcript in such a way that the mauled transcript now has the embedded secret $s \oplus \Delta$, where Δ is sampled by the extractor. The extractor then sends this mauled transcript to the malicious CRS authority. Since this authority offers a service to recover the inputs of the honest parties (or witness in the context of NIZKs), it recovers $s \oplus \Delta$ and outputs this. Note that the authority was tricked into recovering an input that was in reality related to the secret that it hardwired inside the CRS. Now, the extractor has $s \oplus \Delta$, and it can easily recover s and presents it as evidence to implicate the authority. The extractor could never recover s by itself without the "help" of the malicious authority, which also implies defamation free.

While this initial idea sounds promising, its realization involves technical challenges. We highlight some of them below.

- The first and foremost challenge is malleability. We hinged on the fact that the extractor can maul the transcript in the CRS. It turns out that malleability is a challenging problem. Malleability of transcripts has not been studied in the context of interactive protocols before and moreover, even in the setting of NIZKs, this has only been studied in the context of restricted relations.

- Another challenge is to ensure that the malicious authority cannot distinguish whether it is interacting with an extractor, who is trying to incriminate it, or is it interacting with a malicious party who only intends to learn the inputs of the honest parties. In other words, we need an extractor who can produce transcripts that are computationally indistinguishable from the transcripts produced by real protocol executions (not the ones obtained by mauling the CRS).
- We mentioned above that we force the authority to include a transcript in the CRS. How do we ensure that the authority did indeed include a valid transcript in the CRS? The standard solution to employ a NIZK proof cannot work because the authority can violate soundness since it is the one who generates the CRS.
- Finally, to prove the defamation-free property, we need to argue that any probabilistic polynomial time adversary, (no matter how hard it tries) cannot come up with an evidence to implicate an honest authority. In other words, given the transcript in the CRS, it should be computationally infeasible to recover the secret s.

We now show how to implement our main idea, to construct NIZKs, oblivious transfer and secure 2PC with malicious authority security, and in the process we also discuss how to address the above challenges.

2.1 Malicious Authority Security for NIZK

We start by describing the NP relation associated with the proof system.

NP relation. Every instance in this relation is of the form $(C, c_1, \ldots, c_n, b)$, consisting of three components: (1) A boolean circuit $C : \{0,1\}^n \to \{0,1\}$; (2) Committed input $\mathbf{c} = (c_1, \ldots, c_n)$ hiding some bits $\mathbf{x} = (x_1, \ldots, x_n)$ using decommitments $\mathbf{r} = (r_1, \ldots, r_n)$; (3) A bit b satisfying $b = C(x_1, \ldots, x_n)$. The witness is therefore the bits $(\mathbf{x}, \mathbf{r})$. In particular, embedding the commitments in the language guarantees average case hardness, as opposed to regular circuit satisfiability that might have only worst case hardness.

Base proof system. We start with a NIZK proof system and then modify this system to satisfy the desired properties. The proof system is obtained by employing the standard FLS trick [FLS99].

To prove an instance $(C, c_1, \ldots, c_n, b)$ using a witness $\mathbf{x} = (x_1, \ldots, x_n)$ and de-commitments $\mathbf{r} = (r_1, \ldots, r_n)$, we simply use a NIWI proof system in which the prover can show that either it knows the witness $(\mathbf{x}, \mathbf{r})$, or that it knows a seed $\mathsf{s_{in}}$ for some string y that appears in the CRS, i.e., $y = \mathsf{PRG}(\mathsf{s_{in}})$. When the CRS is honestly generated, with overwhelming probability, such a pre-image does not exist, and thus the proof system is sound. Moreover, the simulator can generate an indistinguishable CRS in which $y = \mathsf{PRG}(\mathsf{s_{in}})$ for some trapdoor $\mathsf{s_{in}}$, enabling it to provide proofs without knowing the witnesses.

Accountability. To achieve accountability, we need to provide more information in the CRS. As a warmup, we will include the commitments $\mathsf{cm}_0 = \mathsf{Com}(0; \mathsf{ch}_0)$ and $\mathsf{cm}_1 = \mathsf{Com}(1; \mathsf{ch}_1)$ for random $\mathsf{ch}_0, \mathsf{ch}_1$. To prove accountability, we define an extractor who first samples a circuit C and a string $\mathbf{x} = (x_1, \ldots, x_n)$ according to the distribution of the honest prover, chooses the commitments from the CRS according to $\mathbf{x}$: $(\mathsf{cm}_{x_1}, \ldots, \mathsf{cm}_{x_n})$. Then, the extractor computes a proof π on the instance $(C, (\mathsf{cm}_{x_1}, \ldots, \mathsf{cm}_{x_n}), C(\mathbf{x}))$. The authority, given the instance and the proof, will output the witness $(\mathbf{x}, \mathsf{ch}_{x_1}, \ldots, \mathsf{ch}_{x_n})$. This witness itself serves as an evidence that can be used to incriminate the authority; this is because it can be publicly verified that (x_i, ch_{x_i}) is a valid opening for cm_{x_i} that appears in the CRS. Moreover, just given the CRS, it is computationally infeasible to produce an opening; thus, defamation-free is guaranteed as well.

There are three major issues with this approach. The *first* issue is the following: the authority upon recovering the witness $(\mathbf{x}, \mathsf{ch}_{x_1}, \ldots, \mathsf{ch}_{x_n})$, or even by just viewing the instance $(C, (\mathsf{cm}_{x_1}, \ldots, \mathsf{cm}_{x_n}), C(\mathbf{x}))$ to be opened, will realize that it corresponds to the randomness associated with the commitments in the CRS. So, it will be able to figure out that it is the extractor who submitted the proof. The *second* issue is that it is unclear how the extractor will be able to produce a valid proof on the instance $(C, \mathsf{cm}_{x_1}, \ldots, \mathsf{cm}_{x_n}, C(\mathbf{x}))$. Indeed, the binding property of the commitment scheme and the soundness of the NIWI proof tell us that this should not be possible. Finally, the *third* issue is that we need to verify that the malicious authority included commitments of 0 and 1 only.

- To get around the first issue, we use a rerandomizable commitment scheme. Given commitment to a message m, we can rerandomize this commitment in such a way that randomness of the new commitment information-theoretically hides the randomness used in the old commitment. To see why this is useful, note that the extractor can rerandomize the commitments $(\mathsf{cm}_{x_1}, \ldots, \mathsf{cm}_{x_n})$ to come new commitments $(c_1, \ldots, c_n)$. Now, the randomness recovered by the authority is identically distributed to fresh commitments of $(x_1, ..., x_n)$.

- To get around the second issue, we add to the language of our NIWI a third branch: given a statement $(C, c_1, \ldots, c_n, b)$, the commitments $(c_1, \ldots, c_n)$ were obtained as re-randomizations of the two commitments cm_0 and cm_1 in the CRS, and the extractor has to provide the re-randomization information. Thus, to generate an implicating transcript, the extractor chooses any circuit C and input $\mathbf{x} = (x_1, \ldots, x_n)$ according to the distribution of the honest prover. Moreover, it evaluates $C(x_1, \ldots, x_n) = b$, re-randomizes the commitments $(\mathsf{cm}_{x_1}, \ldots, \mathsf{cm}_{x_n})$ to obtain $(c_1, \ldots, c_n)$. The extractor then gets an instance $(C, c_1, \ldots, c_n, C(\mathbf{x}))$ with a proof π_{NIWI}.

- Finally, to overcome the third issue, the authority will provide four commitments as part of the CRS, $\big((\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_0^1, \mathsf{cm}_1^1)\big)$, and prove using a NIWI proof (which does not require CRS) that one of the branches $(\mathsf{cm}_0^0, \mathsf{cm}_1^0)$ or $(\mathsf{cm}_0^1, \mathsf{cm}_1^1)$ consists of commitments to both bits $(0, 1)$. Before participating in proving any statement with respect to this CRS, one has to check using the proof provided in the CRS that the CRS is correctly computed, i.e., that there is a method to implicate the authority if corrupted.

To argue accountability with the modified CRS, the extractor needs to pick the correct branch to rerandomize. However, it does not know which is the right branch. Instead it samples one of the branches uniformly at random and proceeds. If we had the guarantee that the authority recovered the witness with non-negligible probability then we have the guarantee that the extractor still succeeds in coming up with an incriminating evidence with non-negligible probability.

We can argue defamation-free using the witness-indistinguishability property of the proof in the CRS in conjunction with the hiding property of the commitment scheme.

2.2 Malicious Authority Security for Oblivious Transfer

We now focus our attention on secure two party computation protocols. To gain better intuition and to understand the difficulties we cope with, we start with studying a specific functionality — oblivious transfer. The solutions and techniques developed in addressing this functionality will also be useful in understanding the general case.

As opposed to NIZK which consists of one message and only the prover has some private input (the witness), in oblivious transfer both parties have private inputs. We consider a parallel repetition of 1-out-of-2 bit oblivious transfer, in which the receiver holds a string $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n) \in \{0,1\}^n$, and the sender holds two messages $\mathbf{m}_0 = (m_1^0, \ldots, m_n^0)$, $\mathbf{m}_1 = (m_1^1, \ldots, m_n^1) \in \{0,1\}^n$. Only the receiver receives the output which is $(m_1^{\sigma_1}, \ldots, m_n^{\sigma_n})$. A two-round protocol of oblivious

transfer in the CRS model consists of a CRS generation algorithm GenCRS (run by the authority) that outputs $\mathsf{CRS}_{\mathsf{OT}}$, and two algorithms $\mathsf{OT}_1, \mathsf{OT}_2$ for generating the transcript. The receiver runs $\mathsf{msg}_R = \mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, \boldsymbol{\sigma})$ to obtain the message msg_R from the receiver to the sender, followed by a message $\mathsf{msg}_S = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{m}_0, \mathbf{m}_1, \mathsf{msg}_R)$ from the sender to the receiver. The receiver then makes some local computation to output $(m_1^{\sigma_1}, \ldots, m_n^{\sigma_n})$.

As the functionality hides information for both the receiver $(m_1^{1-\sigma_1}, \ldots, m_n^{1-\sigma_n})$ and the sender $(\boldsymbol{\sigma})$, both parties might come to the malicious authority and ask to open the same transcript, while extracting different information from it. We, therefore, have to discuss two different scenarios and show that in either case, if the authority offers help to either of the two parties, it can be implicated. In the first scenario, which we call *malicious sender*, the sender submits its view to the authority and tries to learn the input of the receiver. We define *malicious receiver* analogously. We follow the same oblivious transfer protocol that is secure against malicious adversaries. As mentioned earlier in our discussion, to achieve the protocol's security, we cannot hope to prevent the malicious authority from making any trapdoors in $\mathsf{CRS}_{\mathsf{OT}}$. Those trapdoors are essential ingredients when proving the simulation security of the protocol. All we can do is to prevent it from using that trapdoor, and specifically from divulging secrets to others.

Dealing with a malicious sender. Following our general template, the authority generates $\mathsf{CRS}_{\mathsf{OT}}$. Moreover, it randomly samples a challenge $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, for some sufficiently long n, and appends $f(\boldsymbol{\sigma})$ to the CRS, where f is a one-way function. Then, we want to embed $\boldsymbol{\sigma}$ in transcript of OT, i.e., add $\mathsf{msg}_R^{\boldsymbol{\sigma}} = \mathsf{OT}_1(\boldsymbol{\sigma})$. Without knowing the receiver's randomness, no one can learn $\boldsymbol{\sigma}$ just from seeing the message $\mathsf{OT}_1(\boldsymbol{\sigma})$, as guaranteed from the receiver's security in the protocol against the malicious sender. This guarantees defamation free. Yet, our goal is give the extractor the ability to maul those transcripts such that if ever opened, we will have a piece of evidence to implicate the authority.

A natural idea is to just complete the transcript with any $\mathbf{m}_0, \mathbf{m}_1$ to obtain $\mathsf{msg}_S = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R^{\boldsymbol{\sigma}}, \mathbf{m}_0, \mathbf{m}_1)$. Then, to come up with the pair $(\mathsf{msg}_S, \mathsf{msg}_R)$ to the authority. But, the authority will refuse to open such a transcript – it can clearly identify that the receiver's secret input is $\boldsymbol{\sigma}$, i.e., the secret challenge it generated! Moreover, it can identify that the message msg_R in the transcript is identical to the message it published in the CRS. We need a stronger method that enables us to complete transcripts to any input of the sender and to maul the receiver's input and re-randomizes it.

To achieve that, we again successfully avoid the issue of malleability using rerandomization and by adding more information in the CRS. Recall that the protocol is a parallel repetition of bit OT, i.e., $\mathsf{OT}_1(\boldsymbol{\sigma}) = \mathsf{OT}_1^{bit}(\sigma_1), \ldots, \mathsf{OT}_1^{bit}(\sigma_n)$, where $(\mathsf{OT}_1^{bit}, \mathsf{OT}_2^{bit})$ is the underlying bit OT protocol. The authority will have to generate for every bit two transcripts, $\alpha_{i,0} = \mathsf{OT}_1^{bit}(\sigma_i \oplus 0)$ and $\alpha_{i,1} = \mathsf{OT}_1^{bit}(\sigma_i \oplus 1)$. This enables the extractor to obtain a transcript for $\boldsymbol{\sigma} \oplus \boldsymbol{\Delta}$ for the receiver for every $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_n)$ of its choice, and any input $(\mathbf{m}_0, \mathbf{m}_1)$ of the sender of its choice. That is, to generate $\mathsf{OT}_1(\boldsymbol{\sigma} \oplus \boldsymbol{\Delta})$, do the following:

$$\mathsf{OT}_1(\boldsymbol{\sigma} \oplus \boldsymbol{\Delta}) = (\alpha_{1,\Delta_1}, \ldots, \alpha_{n,\Delta_n}) = \left(\mathsf{OT}_1^{bit}(\sigma_1 \oplus \Delta_1), \ldots, \mathsf{OT}_1^{bit}(\sigma_n \oplus \Delta_n)\right) .$$

It re-randomizes each one of these messages, and completes it to full transcript with any messages $\mathbf{m}_0, \mathbf{m}_1$ of its choice. The authority receiving such a transcript has no way to tell that this transcript was generated using the transcripts it published in the CRS. By extracting the input of the receiver it discloses itself.

Dealing with a malicious receiver. Following a similar approach, recall that on input $(\mathbf{m}_0, \mathbf{m}_1)$ for the sender and $\mathbf{r}$ for the receiver, the oblivious transfer functionality hides only

$(m_1^{1-r_1}, \ldots, m_n^{1-r_n})$ but reveals $(m_1^{r_1}, \ldots, m_n^{r_n})$ to the receiver. Therefore, it seems natural to embed the challenge in the hidden part of the message.

The authority chooses a new challenge $\mathsf{ch} = (\mathsf{ch}_1, \ldots, \mathsf{ch}_n)$ and publishes $f(\mathsf{ch})$ in the CRS. Moreover, it creates transcripts that correspond to ch and enables the extractor to produce transcript for every input $\mathbf{r} = (r_1, \ldots, r_n)$ of the receiver and a "shift" $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_n)$, while embedding ch_i in position $1 - r_i$ (which is not revealed), to obtain $((\mathsf{ch}_1 \oplus \Delta_1)^{1-r_1}, \ldots, (\mathsf{ch}_n \oplus \Delta_n)^{1-r_n})$. If the malicious authority ever opens the transcript, that is, it recovers $((\mathsf{ch}_1 \oplus \Delta_1)^{1-r_1}, \ldots, (\mathsf{ch}_n \oplus \Delta_n)^{1-r_n})$, then the extractor can extract ch from this (since it knows the "shift", $\boldsymbol{\Delta}$) and implicate the authority.

To do that, first observe that there are 8 possible transcripts for each bit OT: the input of the receiver is a bit $r_i \in \{0, 1\}$ and the input of the sender is one out of four possibilities $((0, 0), (0, 1), (1, 0), (1, 1))$. To enable the extractor to generate any transcript it wishes, for every bit-OT $i \in \{1, \ldots, n\}$, the CRS authority is expected to produce (as part of the CRS generation) four values as follows: For every $m, \Delta \in \{0, 1\}$:

$$\beta^i_{0,m,\Delta} = \mathsf{OT}(0, (m, \mathsf{ch}_i \oplus \Delta)) \quad \text{and} \quad \beta^i_{1,m,\Delta} = \mathsf{OT}(1, (\mathsf{ch}_i \oplus \Delta, m)) \ ,$$

while $\mathsf{OT}(r, (a, b))$ denotes a full transcript of a bit OT where the input of the receiver is $r \in \{0, 1\}$ and the sender is (a, b), and we omit $\mathsf{CRS}_{\mathsf{OT}}$ for brevity. Observe that for each $i \in \{1, \ldots, n\}$ of the bit-OTs provided in the CRS, the bit ch_i is not revealed in the transcript, as it corresponds to the input of the sender that is not revealed. On the other hand, the randomness and the input of the receiver is given in the clear.

The extractor can now choose any message $\mathbf{m} = (m_1, \ldots, m_n)$ of its choice and any $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_n)$, and for every input $\mathbf{r} = (r_1, \ldots, r_n)$ of the receiver, it can generate a transcript $(\beta^1_{r_1,m_1,\Delta_1}, \ldots, \beta^n_{r_n,m_n,\Delta_n})$, which embeds a masking of ch that appears in the CRS. Specifically, for every i, the input of the sender corresponding to r_i is m_i, and the input of the sender corresponding to $1 - r_i$ (the one which is not revealed in the protocol) corresponds to $\mathsf{ch}_i \oplus \Delta_i$. The extractor rerandomizes this transcript and, if opened by the authority, the extractor reconstruct the challenge message ch.

Finally, as the authority has to produce many transcripts that are correlated with the challenge, it has to prove that it generated all of them as specified. Just as in malicious authority security for NIZK, we double all the new information in the CRS and ask it to prove using a NIWI that one of the branches was generated as specified.

Re-randomizable oblivious transfer. As mentioned above, to allow this to work, we must ensure that the oblivious transfer transcript is *rerandomizable*. Informally, we say that an OT transcript is rerandomizable if given a transcript of execution of OT, we should be able to transform into another transcript on the same inputs. The rerandomization guarantee is that even given the secret randomness and the input of both parties in the original transcript, a distinguisher receiving a view of one of the parties should not be able to figure out whether the view comes from a new transcript (with the same inputs), or the view was rerandomized. We show that the oblivious transfer protocol of Peikert, Vaikuntanathan, and Waters [PVW08] is a perfect fit for our needs: it is a two-round oblivious transfer in the CRS model, and we augment the protocol with rerandomization procedures.

Strong accountability. The protocol described above works when the malicious authority does not participate actively in the protocol execution. To understand why, we first remark that for a malicious receiver, the authority provides transcripts where the input and randomness of the receiver are in the clear (i.e., provides the complete view of the receiver).

However, in strong accountability, the extractor now talks directly to the adversary. It receives the first message in the protocol from the adversary, and it cannot know the randomness and the private input of the adversary. Matching the correct transcript from the CRS to the one just received by the adversary is impossible, due to the receiver's privacy. Specifically, to embed a shift Δ_i in position $1 - r_i$, we have to know r_i.

On the other hand, this problem does not occur for the malicious sender's case, as it sends the second message in the protocol. In fact, the above-described method already achieves strong accountability against malicious sender.

Achieving strong accountability via indistinguishability obfuscation. We first start with the following idea. As now the transcripts are given "on the fly" to the extractor, we do not even try to embed the secret challenge into the transcript. Instead, we generate an honestly generated transcript using random messages $\mathbf{m}_0 = (m_1^0, \ldots, m_n^0)$, $\mathbf{m}_1 = (m_1^1, \ldots, m_n^1)$ that the extractor itself *does not even know*, and give it also $\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathsf{ch}$, where ch is the challenge. Now the bits $(m_1^{1-r_1}, \ldots, m_n^{1-r_n})$ are not known to the sender and the receiver, and thus ch is protected using one-time pad.

To implement this idea, we use indistinguishability obfuscation.[2] The authority obfuscates a circuit C that on input msg_R generates msg_S on random inputs $\mathbf{m}_0, \mathbf{m}_1$ and gives also $\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathsf{ch}$. Crucially, the evaluator of the circuit (the extractor) does not know $\mathbf{m}_0, \mathbf{m}_1$. The messages $\mathbf{m}_0, \mathbf{m}_1$ are generated using a pseudorandom key chosen by the CRS authority and was hardwired in the circuit. Moreover, the proof of defamation free is now rather involved as it requires showing that the hardwired challenge ch is not revealed even though it is also hardwired in the circuit.

This description is too simplified and is not sound. If the extraction just sends the message msg_R as received from the obfuscated circuit, the authority can clearly identify it as it had generated the obfuscated circuit. The extractor therefore has to rerandomize the message it receives as output from the circuit. But this still does not suffice, as the authority can also identify that two messages $\mathbf{m}_0, \mathbf{m}_1$ were generated by the circuit, as those are pseudorandom and it knows the key used to generate them. Therefore, we modify the circuit such that given a message msg_R it generates four different transcripts, i.e.,

$$\beta_{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{msg}_R,\ \mathbf{m}_0 \oplus \boldsymbol{\tau}_0,\ \mathbf{m}_1 \oplus \boldsymbol{\tau}_1)$$

for every $\tau_0, \tau_1 \in \{0,1\}$, where $\boldsymbol{\tau} = (\tau, \ldots, \tau)$. This enables the extractor to pick any masking it wishes to $\mathbf{m}_0, \mathbf{m}_1$, similarly to the case of weak accountability. Whenever the authority opens such a transcript, the extractor recovers both $\mathbf{m}_0, \mathbf{m}_1$ and thus also the challenge ch (as it also received $\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathsf{ch}$ from the obfuscated circuit).

2.3 Malicious Authority Security for Two Party Computation

We now focus on general purpose secure two party computation protocols. We first start with an overview that shows that it is not possible to achieve general purpose secure computation protocols for all functions. We then complement our negative result by identifying a class of functionalities (which subsumes oblivious transfer functionalities) for which we can achieve our general positive result.

Impossibility result. When it comes to tackling the problem of designing secure computation for general functionalities, we realize that we cannot simultaneously achieve accountability and defamation-free properties. Since the authority recovers the inputs of the honest parties, it has to

[2] An indistinguishability obfuscator [BGI+01, GGH+16] is a compiler that on input circuit C outputs a functionally equivalent circuit $\widehat{C}$. Moreover, it gurantees that the obfuscations of two functionally equivalent circuits (of the same size) are computationally indistinguishable.

be the case that the output of the functionality does not trivially leak the inputs, i.e., each party really needs the help of the authority to recover the inputs of the honest parties. When considering functions that do not provide such secrecy, we show that there does not exists a protocol that can address malicious authority security. Luckily, those functions, at least intuitively, are the functions for which secure computation is not necessary to begin with. We formalize this intuition and show that it is impossible to achieve secure two party computation for any functionality, as there exists functions for which achieving malicious authority security is impossible.

Observe also that so far, both in the NIZK example and in oblivious transfer, the functionality hides sufficient information (in NIZK this is the witness w; in oblivious transfer these are the choice bits of the receiver and for the sender, or the bits in the input that were not selected by the receiver). This is very intuitive: If the function is not hiding, then there is no need for help from the authority to recover the private inputs.

Positive result. We therefore restrict our attention to a specific class of functions $\mathcal{F}$ that is guaranteed to have some form of secrecy. We also focus on the asymmetric case where only one of the parties (designated as the receiver) gets an output. Specifically we look at functions in which:

1. The inputs of both parties is sufficiently long; and
2. For every input of the receiver, there exists at least λ bits in the inputs of the sender that are not meaningful and do not affect the output, where λ denotes the security parameter.

Interesting functions that are captured in this class are oblivious transfer, private information retrieval, subset sum, and more. We describe a two-round secure computation protocol for computing all the functions in the family in the CRS model, and then show how to enhance its security to achieve malicious authority security.

The base protocol is a standard two-round two-party secure computation protocol that combines two-round oblivious transfer with garbled circuits. Denoting the sender's input as $\mathbf{x} = (x_1, \ldots, x_\ell)$ and the receiver's input as $\mathbf{y} = (y_1, \ldots, y_\ell)$, the receiver's first message is simply $\mathsf{msg}_R = \mathsf{OT}_1(\mathbf{y})$. The second message of the sender is a bit more involved, and this complication comes to accommodate malicious authority security at a later stage. Given a circuit C for computing the function $F \in \mathcal{F}$, the sender generates a garbled circuit GC together with the labels $K^{(x)}_{i,0}, K^{(x)}_{i,1}$ for each input of the sender, and labels $K^{(y)}_{i,0}, K^{(y)}_{i,1}$ for each input of the receiver. To receive the output on $\mathbf{x}, \mathbf{y}$, the receiver has to obtain the labels $K^{(x)}_{i,x_i}, K^{(y)}_{i,y_i}$ and output $\mathsf{Eval}(\mathsf{GC}, K^{(x)}_{i,x_i}, K^{(y)}_{i,y_i})$, i.e., the evaluation of the garbled circuit on the labels $K^{(x)}_{i,x_i}, K^{(y)}_{i,y_i}$. To do that, the "classic" approach is to instruct the sender to send the following message:

$$\{K^{(x)}_{i,x_i}\}_{i\in[\ell]}, \quad \mathsf{OT}_2\left(\mathsf{msg}_R, \ (K^{(y)}_{i,0})_{i\in[\ell]}, (K^{(y)}_{i,1})_{i\in\ell}\right) \ , \tag{1}$$

i.e., sending the labels that correspond to the sender's input, and the labels correspond to the receiver's input are obtained using the oblivious transfer. For the generation of the transcript by the extractor, it would be easier to send the labels of the sender also in an OT message. We instead send the following message:

$$\mathsf{OT}_2\left(\mathsf{msg}_R, \ (K^{(x)}_{i,0} \parallel K^{(y)}_{i,0})_{i\in[\ell]}, \quad (K^{(x)}_{i,x_i} \| K^{(y)}_{i,1})_{i\in\ell}\right) \ . \tag{2}$$

That is, the receiver always receives labels that corresponds to its input, as before. As for the labels that correspond to the input of the sender, in case $y_i = 1$, then the receiver obtains $K^{(x)}_{i,x_i}$, i.e., the "correct" label. On the other hand, in case $y_i = 0$ then the receiver obtains $K^{(x)}_{i,0}$, regardless of what

the input of the sender is, i.e., *it receives a label that corresponds to $x_i = 0$, unlike Eq. (1).* This still guarantees correctness as in the case where $y_i = 0$, the input x_i does not affect the output, and the evaluation of the circuit when $x_i = 0$ and $x_i = 1$ gives the same result (recall requirement 2 in the class of functions for which our possibility result holds). Together with this OT_2 message, the sender also sends a NIWI proof that it either generated the message correctly as instructed, or it knows some trapdoor in the CRS.

Enhancing to malicious authority security. Since the first message in the protocol of the receiver is just $\mathsf{OT}_1(\mathbf{y})$ where $\mathbf{y}$ is the input of the receiver, this case reduces to the case of obtaining malicious authority security in the case of a malicious receiver in oblivious transfer.

For the case of malicious sender, we again follow our general template and the CRS authority chooses a random challenge $\mathsf{ch} = (\mathsf{ch}_1, \ldots, \mathsf{ch}_\lambda) \in \{0,1\}^\lambda$, and gives out $f(\mathsf{ch})$. The goal now is to find what transcripts to provide such that the extractor will be able to embed ch to the input of the sender. If we would have sent the keys as described in Eq. (1), then the authority has to provide information on ch in both $K_{i,0}^{(y)}, K_{i,1}^{(y)}$. However, such transcripts reveal information on the choice bit of the receiver (in our case, we look at the case where $y_i = 0$), and this would break defamation free. Embedding the changes inside the OT_2 message as in Eq. (2) enables us to maul that message without giving away any information about ch. Specifically, we always maul the part that the receiver did not ask for, similarly to the way it is done for oblivious transfer.

3 Preliminaries

Notation and conventions. We let λ denote the security parameter. We let $[n]$ denote the set $\{1, \ldots, n\}$. We use PPT as shorthand for probabilistic polynomial time. A function μ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large λ's, it holds that $\mu(\lambda) < 1/p(\lambda)$.

A probability ensemble $X = \{X(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \{0,1\}^*$ and λ. In the context of zero knowledge, the value a will represent the parties' inputs and λ will represent the security parameter. All parties are assumed to run in time that is polynomial in the security parameter. Two probability ensembles $X = \{X(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$, $Y = \{Y(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$ are are said to be computationally indistinguishable, denoted by $X \approx_c Y$, if for every non-uniform polynomial-time algorithm D there exists a negligible function μ such that for every $a \in \{0,1\}^*$ and every $\lambda \in \mathbb{N}$,

$$|\Pr[D(X(a,\lambda)) = 1] - \Pr[D(Y(a,\lambda)) = 1]| \leq \mu(\lambda)$$

We denote by $x \leftarrow D$ a sampling of an instance x according to the distribution D.

We denote vectors using a **bold** font, e.g., $\boldsymbol{\alpha} \in \{0,1\}^n$, when it is usually clear from context what is the vector size. We let $\boldsymbol{\alpha}[i]$ denote the ith coordinate of the vector.

3.1 Rerandomizable Commitment Scheme

We require commitment scheme that is perfectly binding, computationally hiding, and rerandomizable. The commitment scheme $\mathsf{C} = (\mathsf{Setup}, \mathsf{Com}, , \mathsf{Rerand}, f_{\mathsf{com}})$ has the following syntax and properties:

- $\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda)$: outputs some public parameters p. Let the message space be $\mathcal{M}$ and the commitment space be $\mathcal{C}$.
- $c \leftarrow \mathsf{Com}(\mathsf{p}, m; r)$: The algorithm gets $m \in \mathcal{M}$ and opening $r \in \{0,1\}^{\mathsf{poly}(\lambda)(\lambda)}$, and outputs a commitment $c \in \mathcal{C}$. The opening of the commitment is simply r.

- $c' \leftarrow \mathsf{Rerand}(\mathsf{p}, c; s)$: On input parameters p, commitment c and randomness s, Rerand outputs a randomized commitment c' to the same value. Moreover, we require the existence of an efficient function f_{com} such that for any randomness m, r, s the following holds:
  - $\mathsf{Rerand}(\mathsf{p}, \mathsf{Com}(\mathsf{p}, m; r); s) = \mathsf{Com}(\mathsf{p}, m, s')$ where $s' = f_{\mathsf{com}}(r, s)$.

  Moreover, it is required that for every fixed s, the function $f_{\mathsf{com}}(\cdot, s)$ is bijection, and for every r, the function $f_{\mathsf{com}}(r, \cdot)$ is a bijection as well. In particular, this means that given s', s one can find r for which $s' = f_{\mathsf{com}}(r, s)$.

We require the following from the commitment scheme:
Perfectly binding: For all $(m_0, m_1) \in \mathcal{M}$ such that $m_0 \neq m_1$ and for all $r_0, r_1 \in \{0,1\}^{\mathsf{poly}(\lambda)(\lambda)}$ it holds that:

$$\Pr\left[\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda) \ : \ \mathsf{Com}(\mathsf{p}, m_0; r_0) = \mathsf{Com}(\mathsf{p}, m_1; r_1)\right] = 0 \ .$$

Computational Hiding: Let $\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda)$. For all $(m_0, m_1) \in \mathcal{M}$:

$$\{\mathsf{Com}(\mathsf{p}, m_0)\} \approx_c \{\mathsf{Com}(\mathsf{p}, m_1)\} \ .$$

Such a scheme can be constructed from the DLIN assumption, as showed in [GOS06]. Its rerandomization properties were discussed in [ADKL19].

3.2 Non-Interactive Zero Knowledge (NIZK)

Let L be an NP language and let R_L be its associated relation. For $(x, w) \in R_L$ we sometimes denote x the statement and w its associated witness.

Definition 3.1. *Let $L \in \mathsf{NP}$ and let R_L be the corresponding NP relation. A triple of algorithms $\Pi = (\mathsf{GenCRS}, \mathsf{Prove}, \mathsf{Verify})$ is called* non interactive zero knowledge (NIZK) argument *for L if it satisfies:*

- **Perfect completeness:** *For all security parameters $\lambda \in \mathbb{N}$ and for all $(x, w) \in R_L$,*

$$\Pr\left[\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda);\ \pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, x, w) : \mathsf{Verify}(\mathsf{CRS}, x, \pi) = 1\right] = 1$$

- **Adaptive Computational Soundness:** *For any all PPT prover P^*, there exists a negligible function μ such that for all λ:*

$$\Pr\left[\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda); (x, \pi) \leftarrow P^*(\mathsf{CRS}) : \mathsf{Verify}(\mathsf{CRS}, x, \pi) \wedge x \notin L\right] \leq \mu(\lambda)$$

  When this probability is 0, we say that Π is perfectly sound.
- **Adaptive Zero Knowledge:** *There exists a PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ where $\mathcal{S}_1(1^\lambda)$ outputs $(\mathsf{CRS}_{\mathcal{S}}, \tau)$ and $\mathcal{S}_2(\mathsf{CRS}_S, \tau, x)$ outputs $\pi_{\mathcal{S}}$ such that for all non-uniform PPT adversaries $\mathcal{A}$,*

$$\begin{aligned}&\left\{\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda) \ : \ \mathcal{A}^{\mathcal{O}_1(\mathsf{CRS}, \cdot, \cdot)}(\mathsf{CRS})\right\} \\ \approx_c \ &\left\{(\mathsf{CRS}_{\mathcal{S}}, \tau) \leftarrow \mathcal{S}_1(1^\lambda) \ : \ \mathcal{A}^{\mathcal{O}_2(\mathsf{CRS}, \tau, \cdot, \cdot)}(\mathsf{CRS}_{\mathcal{S}})\right\}\end{aligned}$$

  where $\mathcal{O}_1, \mathcal{O}_2$ on input (x, w) first check that $(x, w) \in R_L$, else output $\perp$. Otherwise, $\mathcal{O}_1$ outputs $\mathsf{Prove}(\mathsf{CRS}, x, w)$ and $\mathcal{O}_2$ outputs $\mathcal{S}_2(\mathsf{CRS}_{\mathcal{S}}, \tau, x)$.

3.3 Non-Interactive Witness Indistinguishability (NIWI)

One building block that we often use in our construction is non-interactive witness indistinguishability. It is useful for our purposes as it does not require a common-reference string. NIWI in the plain model can be constructed based on the DLIN assumption [GOS06] and can be constructed assuming either trapdoor permutations and derandomization assumptions [BOV03].

Definition 3.2. *A pair of PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *is a NIWI for an* NP *relation* R_L *if it satisfies:*

1. **Completeness:** *For every* $(x, w) \in R_L$,

$$\Pr\left[\mathsf{Verify}(x, w) = 1 \ : \ \pi \leftarrow \mathsf{Prove}(x, w)\right] = 1 \ .$$

2. **Soundness:** *There exists a negligible function* μ *such that for every* $x \notin L$ *and* $\pi \in \{0,1\}^*$*:*

$$\Pr\left[\mathsf{Verify}(x, \pi) = 1\right] \leq \mu(|x|) \ .$$

3. **Witness indistinguishability:** *For any sequence* $\{(x, w_1, w_2) : w_1, w_2 \in R_L(x)\} \in \mathcal{I}$*:*

$$\{\pi_1 : \pi_1 \leftarrow \mathsf{Prove}(x, w_1)\}_{(x,w_1,w_2)\in\mathcal{I}} \quad \approx_c \quad \{\pi_2 : \pi_2 \leftarrow \mathsf{Prove}(x, w_2)\}_{(x,w_1,w_2)\in\mathcal{I}} \ .$$

3.4 Secure Two-Party Computation

We recall the definition of secure two-party computation. In this work, we focus on security computation in the CRS model.

Let P_1, P_2 denote the two parties. Each party P_i has a private input x_i and the goal of the parties is to interact with each other to compute $F(x_1, x_2)$, for some functionality $F(\cdot, \cdot)$. The standard simulation-based security guarantee states that even if one of the parties is corrupted by a ppt adversary, then the privacy of the inputs of the honest party is maintained.

To recall the standard notion of simulation security, we first recall the descriptions of the real and the ideal worlds.

Real World. Each party P_i receives has input x_i. A malicious PPT adversary $\mathcal{A}$ is allowed to corrupt some $i \in \{1, 2\}$. Each party computes the next message in the protocol as a function of its private input and randomness, and the messages it received so far. At the end of the protocol, the adversary outputs its view. The output of the this experiment consists of the output of the adversary and the output of the honest party. Denote this output by $\mathsf{Real}_{\Pi,\mathcal{A}}$.

Ideal World. In this world, the ideal adversary (the simulator) corrupts some party $i \in \{1, 2\}$ and let j be the honest party. The trusted party receives x_i form the simulator and P_j sends x_j to the trusted party. The trusted party computes $(y_1, y_2) = F(x_1, x_2; r)$ for a uniform string r and hands the simulator with y_i and P_j with y_j. The output of this world is the output of the simulator and the output of the honest party. Denote this by $\mathsf{Ideal}_{\Pi,\mathsf{Sim}}$.

Definition 3.3 (Simulation Security)**.** *Let* π *and* F *be as above. We say that* π *computes the functionality* F *with* simulation security *if the following holds: for every PPT (malicious) adversary* $\mathcal{A}$ *corrupting any strict subset of the parties, there exists a PPT simulator* Sim *such that the following holds:*

$$\{\mathsf{Ideal}_{\pi,\mathsf{Sim}}\} \approx_c \{\mathsf{Real}_{\pi,\mathcal{A}}\}$$

Hybrid model. An execution of the protocol π in the g-hybrid model, for some functionality g, involves the parties sending normal messages to each other (as in the real model) and, in addition, having access to a trusted party computing g. The composition theorem of [Can00] imply that if π security computes some function f in the g hybrid model, and a protocol ρ computes g, then the protocol π^g (where every ideal call of g is replaced with an execution of ρ) security computes f in the real model.

The common reference string functionality. The common reference string functionality $\mathcal{F}_{\mathsf{CRS}}$ is parameterized with some distribution $\mathcal{C}$ that can be sampled by a PPT algorithm. The functionality is described next.

Functionality 3.4: The $\mathcal{F}_{\mathsf{CRS}}^{\mathcal{C}}$ Functionality

1. Sample $\mathsf{CRS} \leftarrow \mathcal{C}$.
2. Upon receiving a query (P_1, crs) from P_1, send it the value CRS.
3. Upon receiving a query (P_2, crs) from P_2, send it the value CRS.

Looking ahead, the protocols we discuss are in the CRS hybrid model (we call it "in the CRS model" for short). The implementation of this functionality is done using a trusted authority, and the paper addresses the case where this authority is compromised and how it can be implicated in that case.

3.5 Garbled Circuits

We consider a garbled circuit $\mathsf{Garble} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Sim})$ scheme with the following syntax. Given a circuit $C : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$ (as the circuit comes to model a two party function, we explicitly associate input wires with the input of each party, S and R), we define:

- $(\mathsf{GC}, (K_{i,0}^{(S)}, K_{i,1}^{(S)}, K_{i,0}^{(R)}, K_{i,1}^{(R)})_{i=1}^{\ell}) \leftarrow \mathsf{Gen}(1^\lambda, C)$: takes as input a circuit $C : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$, and outputs a garbled circuit GC and set of input wire labels $(K_{i,0}^{(S)}, K_{i,1}^{(S)}, K_{i,0}^{(R)}, K_{i,1}^{(R)})_{i=1}^{\ell}$. For simplicity, the size of all labels is λ.
- $z \leftarrow \mathsf{Eval}(\mathsf{GC}, \{K_{i,x_i}^{(S)}\}_{i\in[\ell]}, \{K_{i,y_i}^{(R)}\}_{i\in\ell})$ The evaluation algorithm takes as input the garbled circuit and the set of labels $\{K_{i,x_i}^{(S)}\}_{i\in[\ell]}, \{K_{i,y_i}^{(R)}\}_{i\in\ell}$ corresponding to inputs $\mathbf{x} = (x_1, \ldots, x_\ell)$ and $\mathbf{y} = (y_1, \ldots, y_\ell)$.
- $(\widetilde{\mathsf{GC}}, \{K_i^{(S)}\}_{i\in[\ell]}, \{K_i^{(R)}\}_{i\in\ell}) \leftarrow \mathsf{Sim}(1^\lambda, C, C(\mathbf{x}, \mathbf{y}))$: takes as input the circuit C and output $C(\mathbf{x}, \mathbf{y})$ and produces a simulated garbled circuit $\widetilde{GC}$ and labels $\{K_i^{(S)}\}_{i\in[\ell]}, \{K_i^{(R)}\}_{i\in\ell}$. Note that the simulator does not know the inputs $\mathbf{x}, \mathbf{y}$, and it generates the input labels according to the output.

We require the following properties from the scheme:

Correctness: We require that for every $\mathbf{x}, \mathbf{y}$ it holds that

$$C(\mathbf{x}, \mathbf{y}) = \mathsf{Eval}(\mathsf{GC}, \{K_{i,x_i}^{(S)}\}_{i\in[\ell]}, \{K_{i,y_i}^{(R)}\}_{i\in\ell}) \ ,$$

where $(\mathsf{GC}, (K_{i,0}^{(S)}, K_{i,1}^{(S)}, K_{i,0}^{(R)}, K_{i,1}^{(R)})_{i=1}^{\ell}) \leftarrow \mathsf{Gen}(1^\lambda, C)$ and $\mathbf{x} = (x_1, \ldots, x_\ell)$ and $\mathbf{y} = (y_1, \ldots, y_\ell)$.

Simulation security: We require the following distributions to be computationally close:

$$\left\{\mathsf{GC}, \{K_{i,x_i}^{(S)}\}_{i\in[\ell]}, \{K_{i,y_i}^{(R)}\}_{i\in\ell}\right\} \approx_c \left\{\widetilde{\mathsf{GC}}, \{\widetilde{K}_i^{(S)}\}_{i\in[\ell]}, \{\widetilde{K}_i^{(R)}\}_{i\in\ell}\right\} ,$$

where GC is generated using Gen for some circuit C and the labels are selected according to inputs $\mathbf{x}, \mathbf{y}$, and $(\widetilde{\mathsf{GC}}, \{\widetilde{K}_i^{(S)}\}_{i\in[\ell]}, \{\widetilde{K}_i^{(r)}\})$ are generated using Sim

3.6 Puncturable Pseudorandom Functions

A pseudorandom function family consisting of functions of the form $\mathsf{PRF}_K(\cdot)$, that is defined over input space $\{0,1\}^*$, output space $\{0,1\}^{\chi(\lambda)}$ and key $K \in \{0,1\}^\lambda$, is said to be a puncturable PRF family if there exists a PPT algorithm $\mathsf{Puncture}$ that satisfies the following properties:

- **Functionality preserved under puncturing:** Puncture takes as input a PRF key $K \in \{0,1\}^\lambda$, input $x \in \{0,1\}^*$ and outputs K_x such that for all $x' \neq x$ it holds that $\mathsf{PRF}_{K_x}(x') = \mathsf{PRF}_{K_x}(x)$
- **Pseudorandom at punctured points:** For every PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1(1^\lambda)$ outputs an input $x \in \{0,1\}^*$, consider an experiment where $K \leftarrow_R \{0,1\}^\lambda$ and $K_x \leftarrow \mathsf{PRF}(K, x)$. Then for all sufficiently large $\lambda \in \mathbb{N}$, for a negligible function μ,

$$\left|\Pr[\mathcal{A}_2(K_x, x, \mathsf{PRF}_K(x)) = 1] - \Pr[\mathcal{A}_2(K_x, x, U_{\chi(\lambda)}) = 1]\right| \leq \mu(\lambda)$$

where $U_\chi(\lambda)$ is a string drawn uniformly at random from $\{0,1\}^{\chi(\lambda)}$.

3.7 Indistinguishability Obfuscation (IO)

An obfuscation scheme associated to a class of circuit $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of two PPT algorithms $\mathsf{iO} = (\mathsf{Obf}, \mathsf{Eval})$ defined below.

- **Obfuscate**, $C' \leftarrow \mathsf{Obf}(1^\lambda, C)$: takes as input security parameter λ, a circuit $C \in \mathcal{C}_\lambda$ and outputs an obfuscation of C, C'.
- **Evaluation**, $y \leftarrow \mathsf{Eval}(C', x)$: a deterministic algorithm that takes as input an obfuscated circuit C', an input $x \in \{0,1\}^\lambda$ and outputs y.

Definition 3.5. *An obfuscation scheme* $\mathsf{iO} = (\mathsf{Obf}, \mathsf{Eval})$ *is* indistinguishable obfuscator *for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$*, with every* $C \in \mathcal{C}_\lambda$ *has size poly*(λ)*, if it satisfies the following properties:*

- **Perfect correctness:** *For every* $C : \{0,1\}^\lambda \to \{0,1\} \in \mathcal{C}_\lambda$*,* $x \in \{0,1\}^\lambda$ *it holds that:*

$$\Pr\left[\mathsf{Eval}\big(\mathsf{Obf}(1^\lambda, C), x\big) = C(x)\right] = 1 .$$

- **Polynomial Slowdown:** *For every* $C : \{0,1\}^\lambda \to \{0,1\} \in \mathcal{C}_\lambda$*, we have the running time of* Obf *on input* $(1^\lambda, C)$ *to be poly*$(|C|, \lambda)$*. Similarly, we have the running time of* Eval *on input* (C', x) *is* $\mathsf{poly}(|C'|, \lambda)$
- **Security:** *For every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$*, such that for every sufficiently large* $\lambda \in \mathbb{N}$*, for every* $C_0, C_1 \in \mathcal{C}_\lambda$ *with* $C_0(x) = C_1(x)$ *for every* $x \in \{0,1\}^\lambda$ *and* $|C_0| = |C_1|$*, we have:*

$$\left|\Pr\left[\mathcal{A}\big(\mathsf{Obf}(1^\lambda, C_0), C_0, C_1\big) = 1\right] - \Pr\left[\mathcal{A}\big(\mathsf{Obf}(1^\lambda, C_1), C_0, C_1\big) = 1\right]\right| \leq \mu(\lambda) .$$

A recent breakthrough work [JLS21] bases iO on well-founded assumptions (see also [BDGM20, WW21, GP21]).

4 Defining Malicious Authority Security

In this section, we define the notion of malicious authority security for cryptographic protocols defined in the CRS model. There are two aspects to our definition, depending on whether CRS is honestly generated or if its generated by a malicious authority. In the first case, if the CRS is honestly generated, we require that the protocol satisfies the same traditional security requirements described in the literature. In the second case, suppose that the CRS is generated by a malicious authority and specifically, if the malicious authority runs a service that let the adversarial entities, participating in the protocol, to recover the inputs of the honest parties. In this setting, we should be able to implicate the malicious authority of its wrongdoing. Formally speaking, we define an extractor that interacts with the malicious authority and comes up with an evidence τ that can presented to a Judge, defined by a Judge algorithm, who verifies whether the presented evidence is valid. At the same time, we require the property that no efficiency adversary can present an evidence that can falsely accuse the honest authority of running a service.

We first discuss the definition of malicious authority security for NIZK (Section 4.1), and then extend the ideas to general secure two party computation (Section 4.2).

4.1 Malicious Authority Security for NIZK

We start by defining malicious authority security in the context of non-interactive zero-knowledge systems.

A NIZK system consists of a triplet of algorithms $\Pi = (\mathsf{GenCRS}, \mathsf{Prove}, \mathsf{Verify})$. In addition, we define a PPT algorithm Judge, which will be necessary for malicious authority security.

- $b \leftarrow \mathsf{Judge}(\mathsf{CRS}, \tau)$ where $b \in \{\mathsf{honest}, \mathsf{corrupted}\}$: The algorithm receives as input the (possibly corrupted) CRS and some transcript τ, and outputs a bit b, indicating whether the τ proves that the CRS CRS is corrupted or not.

Definition 4.1. *Let $L \in \mathsf{NP}$ and let R_L be the corresponding NP relation. We say that a NIZK system $\Pi = (\mathsf{GenCRS}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Judge})$ has malicious authority security with respect to distribution $\mathcal{D}$ if:*

1. *The system $\Pi' = (\mathsf{GenCRS}, \mathsf{Prove}, \mathsf{Verify})$ is a NIZK proof system for L.*
2. **(Accountability:)** *We say that the NIZK scheme Π achieves* accountability with respect to the distribution $\mathcal{D}$ *if for all sufficiently large security parameter $\lambda \in \mathbb{N}$, any* PPT *adversary $\mathcal{A}$, if there exists a non-negligible function $\epsilon_1(\cdot)$ such that*

$$\Pr[\mathsf{Acc.Real}_{\Pi,\mathcal{A},q}(\lambda) = 1] \geq \epsilon_1(\lambda)$$

then there exists a probabilistic polynomial time oracle-aided algorithm Ext making at most q queries, and a non-negligible function $\epsilon_2(\cdot)$ such that:

$$\Pr[\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda) = 1] \geq \epsilon_2(\lambda)$$

where the random variables $\mathsf{Acc.Real}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda)$ and $\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda)$ are defined below.

$\underline{\mathsf{Acc.Real}_{\Pi,\mathcal{A},q}(\lambda)}$*:*
The adversary $\mathcal{A}(1^\lambda)$ outputs some CRS^*, and repeat the following for q iterations:

- $(x_i, w_i) \leftarrow \mathcal{D}$ *and then* $\pi_i \leftarrow \mathsf{Prove}(\mathsf{CRS}^*, (x_i, w_i))$.
- *If* $\mathsf{Verify}(\mathsf{CRS}^*, x_i, \pi_i) \neq 1$ *then abort and the output of the experiment is* 0.
- $\mathcal{A}$ *is given* (x_i, π_i).

$\mathcal{A}$ *outputs some* (i, x_i', w_i') *for* $i \in [q]$. *The output of the experiment is* 1 *if* $x_i = x_i'$ *and* $R_L(x_i, w_i') = 1$.

$\underline{\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda)}$*:*
The adversary $\mathcal{A}(1^\lambda)$ *outputs some* CRS^**, and we invoke the extractor* $\mathcal{E}$ *on input* (CRS^*). *We run* q *iterations in which in each iteration* $\mathcal{E}$ *outputs some* (x_i, π_i) *that is forwarded to* $\mathcal{A}$. *After all iterations,* $\mathcal{A}$ *outputs some* (i, x_i', w_i') *for some* $i \in [q]$. *The extractor* $\mathcal{E}$ *then outputs* τ*, and the output of the experiment is* 1 *if* $\mathsf{Judge}(\mathsf{CRS}^*, \tau) = \mathsf{corrupted}$.

3. **(Defamation-free:)** *For every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$*, such that for all* λ*:*

$$\Pr\left[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}\right] \leq \mu(\lambda) \ ,$$

where $\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda)$.

Discussion. We would like to highlight the following aspects of the above definition:

- **Maliciously generated CRS^*:** In the above definition, the adversary is the one who is choosing the CRS^* which might be maliciously generated. Naturally, our aim is to capture the case where the maliciously generated CRS^* is indistinguishable from an honestly generated one, but the malicious authority has some trapdoors that enable it to extract sensitive information. However, the definition also has to capture the case where CRS^* might be far from an honestly generated CRS, to the extent where the scheme does not even provide correctness with respect to that CRS^*. In that case, the output of the real experiment is 0, and it is easy to implicate the adversary.
- **Oracle access to the adversary:** We remark that the extractor only has an oracle access to the authority and does not get the code of the authority. Giving the code of the authority to the extractor is unrealistic - in real life, the authority is an actual entity, and so the extractor will not have access to its code. Moreover, we want to explicitly prevent the extractor from "rewinding" the authority, as this is not realistic in the real world.
- **The distribution of the queries of the extractor:** The malicious authority $\mathcal{A}$ is the same in both experiments, and as such its view, as generated by the extractor, should be indistinguishable from the view in the real execution. In particular, this means that the malicious authority should receive from $\mathcal{E}$ in each iteration a pair (x_i, π_i) where the marginal distribution on x_i is computationally indistinguishable from the marginal distribution on x_i sampled according to the distribution $(x_i, w_i) \leftarrow \mathcal{D}$ and then setting $\pi_i = \mathsf{Prove}(\mathsf{CRS}^*, x_i, w_i)$.

4.2 Malicious Authority for Secure Two-Party Computation

We now extend the above definition to general two-party computation in the CRS model. We again assume that the reader is familiar with the standard (standalone) definition of secure computation, and refer to the full version for a formal definition. We require that the protocol Π simulates some functionality $\mathcal{F}$ in the CRS model. Then, we add the malicious authority capability. We first provide the definition and then discuss its changes from the NIZK definition. We define a judgement algorithm similarly to the NIZK case:

- $b \leftarrow \mathsf{Judge}(\mathsf{CRS}, \tau)$: It takes as input a common reference string CRS, a certificate τ and outputs a bit b, indicating whether the common reference string CRS is corrupted or not.

Definition 4.2. *We say that a protocol* $\Pi = (\pi, \mathsf{Judge})$ *has malicious authority security for the functionality* $\mathcal{F}$ *with respect to the distribution* $\mathcal{D}$ *if the following conditions hold:*

1. **Simulation security:** *π satisfies simulation security for the functionality $\mathcal{F}$.*
2. **Accountability:** *We say that Π satisfies* security against malicious authority *with respect to the distribution $\mathcal{D}$ if the same conditions as in Definition 4.1 hold, where now the random variables $\mathsf{Acc.Real}_{\Pi,\mathcal{A},q}(\lambda)$ and $\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda)$ are defined as follows.*

   $\mathsf{Acc.Real}_{\Pi,\mathcal{A},q}(\lambda)$*:*
   (a) *$\mathcal{A}(1^\lambda)$ is invoked and outputs a common reference string CRS^*.*
   (b) *The protocol π is executed q number of times, where in the kth execution:*
      - *The adversary $\mathcal{A}$ chooses some input $x_i^{(k)}$ and randomness $r_i^{(k)}$ for P_i.*
      - *The input $x_j^{(k)}$ of the honest party is sampled according to $\mathcal{D}$.*
      - *The protocol is run on these inputs; let trans_k be the resulting transcript.*
      - *The adversary $\mathcal{A}$ receives trans_k.*

   (c) *The adversary outputs $(k, x_j^{(k)})$ for some $k \in [q]$.*
   (d) *The output of the experiment is 1 if the input of P_j in the kth execution was $x_j^{(k)}$.*

   $\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda)$*:*
   (a) *$\mathcal{A}(1^\lambda)$ is invoked and output common reference string CRS^*.*
   (b) *The following is run for q number of times:*
      - *The adversary outputs some input $x_i^{(k)}$ and randomness $r_i^{(k)}$ for P_i.*
      - *The extractor replies with trans_k.*

   (c) *The $\mathcal{A}$ receives the q queries, it outputs $\left(k, x_j^{(k)}\right)$ for some $k \in [q]$. The extractor $\mathcal{E}$ then outputs τ.*
   (d) *The experiment outputs 1 if $\mathsf{Judge}(\mathsf{CRS}^*, \tau) = \mathsf{corrupted}$.*
3. **Defamation free:** *Same as in Definition 4.1.*

Comparison to the malicious authority security of NIZK. Malicious authority security of NIZK is a special case of the above definition for two-party computation. In NIZK, the functionality involves a prover and a verifier, where the prover sends the functionality some (x, w). If $(x, w) \in R_L$ then the functionality sends (x, yes) to the verifier and otherwise it sends (x, no). The protocol in the CRS model consists of a single message from the Prover to the Verifier. As we are interested in the privacy of the witness, we focus on the case where the verifier is corrupted. As a result, the input of the honest party (the prover) is chosen according to the distribution $\mathcal{D}$ and the input of the corrupted party (the verifier) is chosen by the adversary, which is empty in the case of NIZK. The adversary receives the transcript and at some point has to come up with the input of the honest party, i.e., extract from (x_i, π_i) some (x_i, w_i).

4.3 Strong Accountability

So far, we considered adversaries that are passive - while they contribute the input and randomness of the corrupted party in the protocol, the malicious authority expects to get back a full transcript of the protocol, i.e., the adversary is semi-malicious. Here, we model the case where the malicious authority is part of the protocol and colludes with one of the parties. Recall that simulation security is guaranteed only if the CRS authority honestly generated the CRS. If it does not, then we just have accountability as defined next. Defamation free is defined similarly to the previous definitions.

Definition 4.3. *We say that Π satisfies* strong security against malicious authority *with respect to the distribution $\mathcal{D}$ if the conditions as in Definition 4.1 hold, where now the random variables*

$\mathsf{StrongAcc.Real}_{\Pi,\mathcal{A},q}(\lambda)$ *and* $\mathsf{StrongAcc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda)$ *are defined as follows.*

$\underline{\mathsf{StrongAcc.Real}_{\Pi,\mathcal{A},q}(\lambda)}$*:*

1. $\mathcal{A}(1^\lambda)$ *is invoked and outputs a common reference string* CRS^*.
2. *The protocol* π *is executed* q *number of times where in the* k*th execution:*
   - *The adversary* $\mathcal{A}$ *participates in the protocol and corrupts party* P_i.
   - *The input* $x_j^{(k)}$ *of the honest party is sampled according to* $\mathcal{D}$.
3. *The adversary outputs* $(k, x_j^{(k)})$ *for some* $k \in [q]$ *and* $j \neq i$.
4. *The output of the experiment is* 1 *if the input of* P_j *in the* k*th execution was* $x_j^{(k)}$.

$\underline{\mathsf{StrongAcc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda)}$*:*

1. $\mathcal{A}(1^\lambda)$ *is invoked and outputs a common reference string* CRS^*.
2. $\mathcal{A}$ *and* $\mathcal{E}$ *engage in* q *executions of a secure computation protocol, where the party* P_i *is controlled by* $\mathcal{A}$.
3. *After the* q *executions,* $\mathcal{A}$ *outputs* $(k, x_j^{(k)})$ *for some* $k \in [q]$ *and* $j \neq i$. *The extractor* $\mathcal{E}$ *then outputs* τ.
4. *The experiment outputs 1 if* $\mathsf{Judge}(\mathsf{CRS}^*, \tau) = \mathsf{corrupted}$.

5 Malicious Authority Security for NIZK

In this section we construct a NIZK satisfying malicious authority security for the language of circuit satisfiability on committed inputs. The instance is a circuit $C : \{0,1\}^n \to \{0,1\}$, some committed values $\mathbf{c} = (c_1, \ldots, c_n)$ and the output b. The claim is that $\mathbf{c}$ are commitments of bits $\mathbf{x} = (x_1, \ldots, x_n)$, and that $C(x_1, \ldots, x_n) = b$. Formally:

$$R_{\mathsf{p}}(L) = \Big\{(C, c_1, \ldots, c_n, b) \mid \exists (x_1, \ldots, x_n) \in \{0,1\}^m, \quad (r_1, \ldots, r_n) \in \{0,1\}^{\mathsf{poly}(\lambda)} \\ \text{s.t. } C(x_1, \ldots, x_n) = b \\ \wedge \ \forall i \in [n] \ c_i = \mathsf{Com}(\mathsf{p}, x_i; r_i)\Big\}$$

The public parameters p associated with the commitment scheme are part of the CRS. We let $\mathcal{C}$ be the set of all circuits that map n-bit input to 1 bit output.

Tools. The construction is based on the following components:

- A pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\mathsf{poly}(\lambda)}$.
- A rerandomizable bit commitment scheme which is perfectly binding and computationally hiding, denoted as $\mathsf{C} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Rerand})$, as in Section 3.1. The commitment is perfectly binding and computationally hiding.
- Two non-interactive witness-indistinguishable proof systems $(\mathsf{Prove}, \mathsf{Verify})$, denoted as $\Pi^{(1)}_{\mathsf{NIWI}}, \Pi^{(2)}_{\mathsf{NIWI}}$, associated with the languages L_1 and L_2, respectively, which are defined as follows:
  - **The language L_1:**

$$\textbf{Statement}: \quad \big(\mathsf{p}, (\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_0^1, \mathsf{cm}_1^1)\big)$$
$$\textbf{Witness}: \quad \big((r_{\mathsf{Setup}}, \mathsf{ch}_0^0, \mathsf{ch}_1^0), (r_{\mathsf{Setup}}, \mathsf{ch}_0^1, \mathsf{ch}_1^1)\big) \ ,$$

    such that *one* of the following conditions hold:

1. $\mathsf{p} = \mathsf{Setup}(1^\lambda; r_{\mathsf{Setup}})$ and $(\mathsf{cm}_0^0, \mathsf{cm}_1^0) = (\mathsf{Com}(\mathsf{p}, 0, \mathsf{ch}_0^0), \mathsf{Com}(\mathsf{p}, 1, \mathsf{ch}_1^0))$, or
2. $\mathsf{p} = \mathsf{Setup}(1^\lambda; r_{\mathsf{Setup}})$ and $(\mathsf{cm}_0^1, \mathsf{cm}_1^1) = (\mathsf{Com}(\mathsf{p}, 0, \mathsf{ch}_0^1), \mathsf{Com}(\mathsf{p}, 1, \mathsf{ch}_1^1))$.

– **The language L_2:**

$$\textbf{Statement}: \quad (\mathsf{CRS}, C, c_1, \ldots, c_n, b)$$
$$\textbf{Witness}: \quad \left(\mathsf{s}_{\mathsf{in}}, \{x_i\}_{i\in[n]}, \{r_i\}_{i\in[n]}, \sigma, \{s_i\}_{i\in[n]}\right)$$

such that *one* of the following conditions hold:

1. $\mathsf{PRG}(\mathsf{s}_{\mathsf{in}}) = \mathsf{s}_{\mathsf{out}}$, or,
2. $C(x_1, \ldots, x_n) = b$, and $\forall i \in [n]$ it holds that $c_i = \mathsf{Com}(\mathsf{p}, x_i; r_i)$.
3. $C(x_1, \ldots, x_n) = b$, and $\forall i \in [n]$ it holds that $c_i = \mathsf{Rerand}(\mathsf{p}, \mathsf{cm}_{x_i}^\sigma; s_i)$.

where $\mathsf{s}_{\mathsf{out}}$ and $(\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_1^0, \mathsf{cm}_1^1)$ are taken from the CRS as given in Step 5 of GenCRS in Construction 5.1.

Construction 5.1: NIZK with Malicious Authority Security

$\mathsf{GenCRS}\ (1^\lambda)$:

1. Compute $\mathsf{p} \leftarrow \mathsf{Setup}(1^\lambda; r_{\mathsf{Setup}})$ for a random r_{Setup}.
2. Sample $\mathsf{s}_{\mathsf{out}} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
3. For $\sigma = 0$ and $\sigma = 1$, do the following:
   - Compute $(\mathsf{cm}_0^\sigma, \mathsf{cm}_1^\sigma) = (\mathsf{Com}(\mathsf{p}, 0, \mathsf{ch}_0^\sigma), \mathsf{Com}(\mathsf{p}, 1, \mathsf{ch}_1^\sigma))$, for $\mathsf{ch}_0^\sigma, \mathsf{ch}_1^\sigma \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
4. Compute $\pi^{(1)} \leftarrow \Pi_{\mathsf{NIWI}}^{(1)}.\mathsf{Prove}\left(\left(\mathsf{p}, (\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_0^1, \mathsf{cm}_1^1)\right), ((r_{\mathsf{Setup}}, \mathsf{ch}_0^0, \mathsf{ch}_1^0), (\bot, \bot, \bot)\right)$.
5. Output $\mathsf{CRS} = \left(\mathsf{p}, \mathsf{s}_{\mathsf{out}},\ (\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_0^1, \mathsf{cm}_1^1),\ \pi^{(1)}\right)$.

$\mathsf{Prove}(\mathsf{CRS}, (C, c_1, \ldots, c_n, b), \mathbf{w})$: On input CRS, circuit $C : \{0,1\}^m \to \{0,1\}$, commitments $(c_1, \ldots, c_n)$, output bit $b \in \{0,1\}$, and witness $\mathbf{w}$ do the following:

1. Parse $\mathbf{w} = ((x_1, \ldots, x_n), (r_1, \ldots, r_n))$ where $x_i \in \{0,1\}, r_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$ for all $i \in [n]$, and compute

$$\pi^{(2)} \leftarrow \Pi_{\mathsf{NIWI}}^{(2)}.\mathsf{Prove}\left(\ (\mathsf{CRS}, C, c_1, \ldots, c_n, b),\ (\bot, \{x_i\}_{i\in[n]}, \{r_i\}_{i\in[n]}, \bot, \bot)\ \right)\ .$$

Output $\pi = \pi^{(2)}$.

$\mathsf{Verify}(\mathsf{CRS}, (C, c_1, \ldots, c_n, b), \pi)$: On input CRS, instance $(C, c_1, \ldots, c_n, b)$, proof π,

1. Output the decision of $\Pi_{\mathsf{NIWI}}^{(2)}.\mathsf{Verify}((\mathsf{CRS}, C, c_1, \ldots, c_n, b), \pi)$.

$\mathsf{Judge}(\mathsf{CRS}^*, \tau)$: On input (possibly maliciously generated) CRS^* and a transcript τ, do the following:

1. Check that the CRS^* is well formed. That is:
   (a) Parse $\mathsf{CRS}^* = \left(\mathsf{p}, \mathsf{s}_{\mathsf{out}},\ (\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_0^1, \mathsf{cm}_1^1),\ \pi^{(1)}\right)$.
   (b) Verify that $\Pi_{\mathsf{NIWI}}^{(1)}.\mathsf{Verify}\left(\left(\mathsf{p}, (\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_0^1, \mathsf{cm}_1^1)\right), \pi^{(1)}\right) = 1$.
   If the verification fails then abort and output $\mathsf{corrupted}$.
2. If $\tau = (\mathsf{open}, \sigma, \rho, \mathsf{ch})$, where $\sigma, \rho \in \{0,1\}$ and $\mathsf{ch} \in \{0,1\}^{\mathsf{poly}(\lambda)}$ then: If $c_\rho^\sigma = \mathsf{Com}(\mathsf{p}, \rho; \mathsf{ch})$ then output $\mathsf{corrupted}$.
3. Otherwise, output honest.

The distribution $\mathcal{D} = (\mathcal{D}_{\mathcal{C}}, \mathcal{D}_{\mathbf{x}})$. We define now the family of distributions $\mathcal{D}$ associated with the accountability property. We only place a restriction on the generation of commitments; that is, the commitments need to be honestly generated, i.e., the randomness r_i is uniformly distributed in $\{0,1\}^{\mathsf{poly}(\lambda)}$.

Formally, for any distribution $\mathcal{D}_{\mathcal{C}}$ over $\mathcal{C}$ (recall that $\mathcal{C}$ denotes the set of all circuits with m-bit input and 1-bit output) and for any distribution $\mathcal{D}_{\mathbf{x}}$ over $\{0,1\}^m$, the distribution $\mathcal{D}(\mathcal{D}_{\mathcal{C}}, \mathcal{D}_{\mathbf{x}})$, parameterized also by p, samples the input and witness as follows:

1. Sample a circuit C according to $\mathcal{D}_{\mathcal{C}}$.
2. Sample the bits $(x_1, \ldots, x_n)$ according to $\mathcal{D}_{\mathbf{x}}$ and evaluate $b = C(x_1, \ldots, x_n)$.
3. For every $i \in [n]$ compute $c_i = \mathsf{C.Com}(\mathsf{p}, x_i; r_i)$ for a uniform r_i.
4. Output $(C, c_1, \ldots, c_n, b)$ as the instance and $(x_1, \ldots, x_n), (r_1, \ldots, r_n)$ as the witness.

Theorem 5.2. *Assuming the security of* PRG, Com, $\Pi^{(1)}_{\mathsf{NIWI}}$ *and* $\Pi^{(2)}_{\mathsf{NIWI}}$, *Construction 5.1 is a NIZK proof system with malicious authority security for the language* $R_{\mathsf{p}}(L)$ *with respect to any distribution* $\mathcal{D} = (\mathcal{D}_{\mathcal{C}}, \mathcal{D}_{\mathbf{x}})$, *where* p *is sampled as part of* GenCRS.

Proof. We show completeness, soundness, zero-knowledge, accountability and defamation-free properties.

Completeness. The completeness property follows from the completeness property of $\Pi^{(2)}_{\mathsf{NIWI}}$.

Soundness. Let $\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda)$, and fix any PPT adversary (corrupted prover) $\mathcal{A}$ and any $(C, c_1, \ldots, c_n, b) \notin R_{\mathsf{p}}(L)$. We show that the probability that the adversary outputs a proof π such that $\mathsf{Verify}(\mathsf{CRS}, (C, c_1, \ldots, c_n, b), \pi) = 1$ is negligible.

We claim that $(\mathsf{CRS}, C, c_1, \ldots, c_n, b) \notin L_2$ (recall that L_2 is the associated language of $\Pi^{(2)}_{\mathsf{NIWI}}$). Once we prove this, it then follows from the perfect soundness of $\Pi^{(2)}_{\mathsf{NIWI}}$ that the probability that the adversary outputs a valid proof is negligible.

To show this, it suffices to argue that there does not exist any witness $(\mathsf{s_{in}}, (x_1, \ldots, x_n), (r_1, \ldots, r_n), \sigma, \{s_i\}_{i \in [n]})$ for the instance $(\mathsf{CRS}, C, c_1, \ldots, c_n, b)$.

1. Firstly, with overwhelming probability it holds that $\mathsf{PRG}(\mathsf{s_{in}}) \neq \mathsf{s_{out}}$ since $\mathsf{s_{out}}$ is sampled uniformly at random.
2. Since $(C, c_1, \ldots, c_n, b) \notin R_{\mathsf{p}}(L)$, it holds that either $C(x_1, \ldots, x_n) \neq b$, or there exists some $i \in [n]$ such that $c_i \neq \mathsf{Com}(\mathsf{p}, x_i; r_i)$.
3. Finally, we argue that there exists some $i \in [n]$ such that $\mathsf{Rerand}(\mathsf{p}, \mathsf{cm}^{\sigma}_{x_i}; s_i) \neq c_i$. Observe that if c_i was obtained as a rerandomization of some $\mathsf{cm}^{\sigma}_{x_i}$, then still there exists a unique opening to x_i. This would then violate $(C, c_1, \ldots, c_n, b) \notin R_{\mathsf{p}}(L)$.

Zero-Knowledge. We describe a PPT simulator Sim. The simulator Sim runs GenCRS, but compute s_{out} as an output of PRG where the seed $\mathsf{s_{in}}$ is sampled uniformly at random from $\{0,1\}^\lambda$ and stores the trapdoor $\mathsf{s_{in}}$. Then, whenever it is given an instance $(\mathsf{CRS}, C, c_1, \ldots, c_n, b)$ it uses $(\mathsf{s_{in}}, \bot, \bot, \bot, \bot)$ as a witness to compute the proof $\pi^{(2)}$.

The computational indistinguishability of the real world and the ideal world follows from the witness-indistinguishability of $\Pi^{(2)}_{\mathsf{NIWI}}$, and the pesudorandomness of PRG.

Accountability. We first describe the extractor $\mathcal{E}$. On input (possibly maliciously generated) CRS^* generated by the malicious authority, it does the following:

1. It checks that CRS^* is well formed, as described in Step 1 in Judge algorithm. If CRS^* is not well formed, then it halts and outputs $\tau = \bot$.
2. It chooses a branch $\sigma \in \{0,1\}$ uniformly at random and runs the following for q iterations. For every $j \in [q]$:
   (a) It samples a circuit $C^{(j)} = C$ according to the distribution $\mathcal{D}_{\mathcal{C}}$.
   (b) It samples an input $\mathbf{x}^{(j)} = \mathbf{x} = (x_1, \ldots, x_n)$ according to the distribution $\mathcal{D}_{\mathbf{x}}$.
   (c) It evaluates $b^{(j)} = b = C(x_1, \ldots, x_n)$.
   (d) For every $i \in [n]$, it generates the commitment $\hat{c}_i = \mathsf{C.Rerand}(\mathsf{p}, \mathsf{cm}^{\sigma}_{x_i}, s_i)$ for a uniformly random $s_i^{(j)} = s_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Let $(\hat{c}_1, \ldots, \hat{c}_n)$ be the sequence of all the re-randomized commitments.
   (e) It generates $\pi = \pi^{(2)} \leftarrow \Pi^{(2)}_{\mathsf{NIWI}}.\mathsf{Prove}\left((\mathsf{CRS}^*, C, \hat{c}_1, \ldots, \hat{c}_n, b), (\bot, \bot, \bot, \sigma, \{s_i\}_{i\in[n]})\right)$.
   (f) It sends $((C, \hat{c}_1, \ldots, \hat{c}_n, b), \pi)$ to the malicious authority.
3. After q iterations, the malicious authority might reply with some input $(j, (C, \hat{c}_1, \ldots, \hat{c}_n, b))$ with witness $\{x_i\}_{i\in[n]}$ and $\{r_i\}_{i\in[n]}$ for which $\hat{c}_i = \mathsf{Com}(\mathsf{p}, x_i; r_i)$ for all $i \in [n]$ and $C(x_1, \ldots, x_n) = b$.
   Let $s_1^{(j)}$ be the randomness used to re-randomize $\mathsf{cm}^{\sigma}_{x_1}$. Extract $\mathsf{ch}^{\sigma}_{x_1}$ using f_{com} (i.e., find $\mathsf{ch}^{\sigma}_{x_1}$ for which $r_1 = f_{\mathsf{com}}(\mathsf{ch}^{\sigma}_{x_1}, s_1^{(j)})$, see Section 3.1). If it holds that $\mathsf{cm}^{\sigma}_{x_1} = \mathsf{Com}(\mathsf{p}, x_1; \mathsf{ch}^{\sigma}_{x_i})$ then output $\tau = (\mathsf{open}, \sigma, x_1, \mathsf{ch}^{\sigma}_{x_i})$.

We claim that if the extractor chooses σ to be the "correct branch", i.e., the one for which $\pi^{(1)}$ from CRS^* is correct, then the view of the malicious authority is indistinguishable between $\mathsf{Acc.Real}$ and $\mathsf{Acc.Ext}$. We then claim that the extractor always outputs a transcript that is accepted by Judge if the malicious authority outputs a valid witness for some $(j, (C, \hat{c}_1, \ldots, \hat{c}_n, b))$ in the jth iteration (for $j \in [q]$). To show this, we need to argue that the view of the authority when interacting with the real parties is computationally indistinguishable from the view of the authority when interacting with the extractor.

Indistinguishability of views. Consider the following hybrids:

- $\mathbf{H}_1$: This is the experiment $\mathsf{Acc.Real}_{\Pi,\mathcal{A},q}(\lambda)$.
- $\mathbf{H}_2$: This hybrid is inefficient. The prover in the real experiment works as follows: It samples $C \leftarrow \mathcal{D}_{\mathcal{C}}$ and $\mathbf{x} \leftarrow \mathcal{D}_{\mathbf{x}}$. Then, instead of directly committing to the $\mathbf{x} = (x_1, \ldots, x_n)$ it takes $\mathsf{cm}^{\sigma}_{x_1}, \ldots, \mathsf{cm}^{\sigma}_{x_n}$. It then re-randomizes all the commitments to obtain rerandomized commitments $\hat{c}_1, \ldots, \hat{c}_n$. It then runs in exponential time and determines the randomness $\{r_i\}_{i\in[n]}$ such that $\hat{c}_i^{\sigma} = \mathsf{Com}(\mathsf{p}, x_i; r_i)$. It uses $(\bot, \mathbf{x}, \{r_i\}_{i\in[n]}, \bot, \bot)$ to compute the proof $\pi^{(2)}$ for the instance $(C, \hat{c}_1, \ldots, \hat{c}_n, b)$, where $b = C(\mathbf{x})$. The rest of the hybrid is the same as before.
- $\mathbf{H}_3$: This is $\mathsf{Acc.Ext}_{\Pi,\mathcal{A},q}(\lambda)$ with branch σ.

The view of the adversary in $\mathbf{H}_1$ and $\mathbf{H}_2$ is identical, which follows from the fact that the rerandomization procedure of the commitment scheme generates commitments that are identically distributed to fresh commitments.

The two hybrids $\mathbf{H}_2$ and $\mathbf{H}_3$ are computationally indistinguishable, follows from the witness-indistinguishability of $\Pi^{(2)}_{\mathsf{NIWI}}$. We consider a non-uniform reduction which gets as input the CRS and the decommitments of the commitments in the CRS as non-uniform advice. Using this, the (efficient) reduction computes and sends the instance $(C, \hat{c}_1, \ldots, \hat{c}_n, b)$, where $b = C(x_1, \ldots, x_n)$, along with two witnesses to the challenger of the WI game. The challenger

then sends the proof $\pi^{(2)}$ to the reduction, who forwards it to the adversary. If the adversary can distinguish the hybrids $\mathbf{H}_3$ and $\mathbf{H}_4$ with non-negligible probability then the reduction can break the witness-indistinguishability property with non-negligible probability, a contradiction.

From the perfectly binding property of the commitment scheme, the only witness that the authority can give is the same input $(x_1, \ldots, x_n)$ as the extractor used. Thus, when the authority gives $\{x_i\}_{i\in[n]}$ and $\{r_i\}_{i\in[n]}$ for the jth execution, from inspection it is easy to see that the extractor outputs a transcript that implicates the authority. Thus, if the authority succeeds in $\mathsf{Acc.Real}$ with some non-negligible probability $\epsilon_1(\lambda)$ then the extractor succeeds with probability negligibly close to $\epsilon_1(\lambda)/2$, where the loss occurs from guessing σ and the indistinguishability of the proof π.

Defamation-Free. For every PPT adversary $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that: $\Pr[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}] \leq \mu(\lambda)$, where $\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda)$. First, since CRS is honestly generated, it always passes the verification of Step 1 in the Judge algorithm. We now show that for every $\sigma \in \{0,1\}$, no PPT adversary can output $\tau = (\mathsf{open}, \sigma, \cdot)$ with non-negligible probability. For that, fix $\sigma \in \{0,1\}$, and consider the following sequence of hybrid experiments:

1. $\mathbf{H}_1$: In this hybrid, the adversary receives $\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda)$ and the output of the hybrid is 1 if it outputs $\tau = (\mathsf{open}, \sigma, \cdot)$ that is accepted by Judge.
2. $\mathbf{H}_2$: We modify the way CRS is generated, such that for proving the instance $(\mathsf{p}, (\mathsf{cm}_0^0, \mathsf{cm}_1^0), (\mathsf{cm}_0^1, \mathsf{cm}_1^1))$ using $\Pi^{(1)}_{\mathsf{NIWI}}$, we use the witness of $1-\sigma$. That is,
   (a) If $\sigma = 0$ we prove $\pi^{(1)}$ using the witness $((\bot,\bot,\bot),(r_{\mathsf{Setup}}, \mathsf{ch}_0^1, \mathsf{ch}_1^1))$.
   (b) If $\sigma = 1$ we prove $\pi^{(1)}$ using the witness $((r_{\mathsf{Setup}}, \mathsf{ch}_0^0, \mathsf{ch}_1^0),(\bot,\bot,\bot))$.

Clearly, the views of the adversary in both the experiments are computationally indistinguishable from the witness-indistinguishability property of $\Pi^{(1)}_{\mathsf{NIWI}}$. Next, we claim that the probability that the adversary outputs an accepted transcript in $\mathbf{H}_2$ is negligible. At this point, the proof $\pi^{(1)}$ is independent of the randomness used to create the commitments $\mathsf{cm}_0^\sigma, \mathsf{cm}_1^\sigma$. An adversary that outputs $\tau = (\mathsf{open}, \sigma, \rho, \mathsf{ch})$ can be used for violating the hiding property of the commitment scheme. □

6 Malicious Authority Security for Oblivious Transfer

In this section, we show how to achieve malicious authority security for oblivious transfer. We start with formally defining the OT functionality (Section 6.1), and then the rerandomization properties that we need from the basic construction. We show an instantiation of such rerandomizable OT in Section 6.3, and proceed to our two constructions: We show *weak* accountability in Section 6.4, and *strong* accountability in 6.5.

6.1 The OT Functionality

We define a n-out-of-$2n$ OT functionality as follows:

Functionality 6.1: The $\mathcal{F}_{n\times\mathsf{OT}}$ functionality

The functionality interacts with a sender and a receiver.

1. Upon receiving a message $(\mathsf{sender}, \mathbf{m}_0, \mathbf{m}_1)$ from the sender, where $\mathbf{m}_0 = (m_1^0, \ldots, m_n^0) \in \{0,1\}^n$ and $\mathbf{m}_1 = (m_1^1, \ldots, m_n^1) \in \{0,1\}^n$, store $\mathbf{m}_0, \mathbf{m}_1$.

2. Upon receiving a message $(\mathsf{receiver}, \boldsymbol{\sigma})$ with $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n) \in \{0,1\}^n$, check if a message $(\mathsf{sender}, \cdot, \cdot)$ was perviously sent. If yes, then send $(m_1^{\sigma_1}, \ldots, m_n^{\sigma_n})$. If not, send $\perp$ to the receiver.

In the case when $n = 1$, i.e., 1-out-of-2 oblivious transfer, we denote the functionality to be $\mathcal{F}_{\mathsf{OT}}$.

6.2 Two-Round Rerandomizable OT Protocol

We define a notion of rerandomizable OT protocol in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model.

Syntax. A two-round, oblivious transfer protocol in the CRS model (see Functionality 3.4) consists of algorithms $(\mathsf{GenCRS}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{Output})$ and has the following syntax:

1. $\mathsf{CRS}_{\mathsf{OT}} \leftarrow \mathsf{GenCRS}(1^\lambda)$: The algorithm generates the common reference string $\mathsf{CRS}_{\mathsf{OT}}$.
2. $\mathsf{msg}_R \leftarrow \mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, \boldsymbol{\sigma}; \boldsymbol{\rho}_R)$: The algorithm OT_1 receives as input the input $\boldsymbol{\sigma}$ and randomness $\boldsymbol{\rho}_R$. It generates some message msg_R to be sent to the sender.
3. $\mathsf{msg}_S \leftarrow \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{m}_0, \mathbf{m}_1, \mathsf{msg}_R; \boldsymbol{\rho}_S)$: the algorithm OT_2 is invoked with inputs $\mathbf{m}_0, \mathbf{m}_1$ and the message msg_R sent by the receiver to the sender. It generates the next message msg_S to be sent to the receiver, using randomness $\boldsymbol{\rho}_S$.
4. $\mathsf{Output}(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_S; \boldsymbol{\rho}_R)$: Given msg_S received by the sender, and the randomness $\boldsymbol{\rho}_R$ (that encodes the secret state), the receiver computes its output.

Notation. We denote by $\mathsf{trans} = \mathsf{OT}(\mathsf{CRS}_{\mathsf{OT}}, (\boldsymbol{\sigma}; \boldsymbol{\rho}_R), (\mathbf{m}_0, \mathbf{m}_1; \boldsymbol{\rho}_S))$ an honest execution of the protocol with respect to a fixed CRS, where the receiver uses input $\boldsymbol{\sigma}$ and randomness $\boldsymbol{\rho}_R$ and the sender uses input $(\mathbf{m}_0, \mathbf{m}_1)$ and randomness $\boldsymbol{\rho}_S$. We sometimes omit the specification of the random tapes and simply write $\mathsf{trans} \leftarrow \mathsf{OT}(\mathsf{CRS}, \boldsymbol{\sigma}, (\mathbf{m}_0, \mathbf{m}_1))$. We also sometimes omit the specification of the CRS, if the string CRS being used is clear from the context.

Rerandomization Algorithms. Associated with a rerandomizable OT protocol Π_{OT} are three rerandomization algorithms. Roughly speaking, the first algorithm shows how to rerandomize the first message from the receiver to the sender. The second shows how to rerandomize the second message, and the third essentially rerandomizes the entire transcript. The syntax is as follows:

- $\mathsf{msg}'_R \leftarrow \mathsf{Rerand}_1(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R; R)$ Given a first message $\mathsf{msg}_R \leftarrow \mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, \boldsymbol{\sigma})$ for some $\mathsf{CRS}_{\mathsf{OT}}, \boldsymbol{\sigma}$, and random tape R, the algorithm outputs some message msg'_R.
- $\mathsf{msg}'_S \leftarrow \mathsf{Rerand}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R, \mathsf{msg}_S; S)$ Given a message msg_R from the receiver to the sender, and then a message msg_S from the sender to the receiver and a random tape S, the algorithm outputs some new messages msg'_S.
- $\mathsf{trans}' \leftarrow \mathsf{Rerand}_{\mathsf{OT}}(\mathsf{trans}; R, S)$: Given a transcript trans that was generated as $\mathsf{trans} = \mathsf{OT}(\mathsf{CRS}, (\boldsymbol{\sigma}; \boldsymbol{\rho}_R), ((\mathbf{m}_0, \mathbf{m}_1); \boldsymbol{\rho}_S)$, and two random tapes R, S, output a transcript $\mathsf{trans}' = \mathsf{OT}(\mathsf{CRS}, (\boldsymbol{\sigma}; \boldsymbol{\rho}'_R), (\mathbf{m}_0, \mathbf{m}_1; \boldsymbol{\rho}'_S))$ corresponding to some randomness $\boldsymbol{\rho}'_S, \boldsymbol{\rho}'_R$.

We now present the definition of a two-round rerandomizable OT protocol below. In addition to requiring simulation-basesd security, we additionally require that the OT protocol satisfies rerandomizability property.

Definition 6.2. *Let Π_{OT} be a two-round oblivious transfer protocol, and let $\mathsf{Rerand}_1, \mathsf{Rerand}_2$ be rerandomization algorithms as defined above. We say that the protocol Π_{OT} is two-round rerandomizable OT protocol if:*

1. **Simulation Security.** Π_{OT} *securely implements the* $\mathcal{F}_{n\times\mathsf{OT}}$ *functionality in the* $\mathcal{F}_{\mathsf{CRS}}$*-hybrid model (see Functionality 3.4).*
2. **Rerandomizability.** *For every* $\mathsf{CRS}_{\mathsf{OT}} \leftarrow \mathsf{GenCRS}(1^\lambda)$*, for every* $\mathbf{m}_0, \mathbf{m}_1, \boldsymbol{\sigma}$*:*
   (a) *Let* $\mathsf{msg}_R = \mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, \boldsymbol{\sigma}; \boldsymbol{\rho}_R)$ *for some fixed* $\boldsymbol{\rho}_R$*. Then, for every* $\boldsymbol{\rho}_R$ *there exists a bijection* $g_{\boldsymbol{\rho}_R}(\cdot)$*, such that for every* R*:*

$$\mathsf{Rerand}_1\left(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R; R\right) = \mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, \boldsymbol{\sigma}; g_{\boldsymbol{\rho}_R}(R)) \ .$$

   (b) *Let* $\mathsf{msg}_S = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{m}_0, \mathbf{m}_1, \mathsf{msg}_R; \boldsymbol{\rho}_S)$*. Then, for every* $\boldsymbol{\rho}_S$ *there exists a bijection* $g_{\boldsymbol{\rho}_S}(\cdot)$ *such that for every* S*:*

$$\mathsf{Rerand}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R, \mathsf{msg}_S; S) = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{m}_0, \mathbf{m}_1, \mathsf{msg}_R; g_{\boldsymbol{\rho}_S}(S)) \ .$$

   (c) *Let* $\mathsf{trans} = \mathsf{OT}(\mathsf{CRS}_{\mathsf{OT}}, (\boldsymbol{\sigma}; \boldsymbol{\rho}_R), (\mathbf{m}_0, \mathbf{m}_1; \boldsymbol{\rho}_S))$*. Then, for every* $\boldsymbol{\rho}_R, \boldsymbol{\rho}_S$ *there exists a bijection* $g_{\boldsymbol{\rho}_R}(\cdot)$*,* $g_{\boldsymbol{\rho}_S}(\cdot)$*, and:*

$$\mathsf{Rerand}_{\mathsf{OT}}(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{trans}; R, S) = \mathsf{OT}(\mathsf{CRS}_{\mathsf{OT}}, (\boldsymbol{\sigma}; g_{\boldsymbol{\rho}_R}(R)), (\mathbf{m}_0, \mathbf{m}_1; g_{\boldsymbol{\rho}_S}(S))) \ .$$

6.3 Instantiation: The PVW OT Protocol

We now show that the two-round OT protocol of Peikert, Vaikuntanathan and Waters [PVW08], based on the DDH assumption, is rerandomizable and satisfies Definition 6.2. We first review the protocol. For simplicity, we consider $1\times\mathsf{OT}$, i.e., 1-out-of-2 oblivious transfer where the sender holds two bits m_0, m_1, the receiver holds σ and is supposed to learn m_σ. Generalizing to $n\times\mathsf{OT}$ is done via a standard parallel repetition.

In a nutshell, the protocol is as follows: From the CRS and the choice bit σ, the receiver derives some public key pk and a corresponding secret key sk. It sends the public key to the sender. The sender derives two public keys pk_0, pk_1 from pk, and encrypts m_0 using pk_0 and m_1 using pk_1. The security guarantee is that the receiver can only decrypt messages that are encrypted under pk_σ, and that the public key $pk_{1-\sigma}$ is "messy", namely, for every two messages m_0, m_1 in the message space it holds that the two distributions $\{\mathsf{Enc}_{pk_{1-\sigma}}(m_0)\}$ and $\{\mathsf{Enc}_{pk_{1-\sigma}}(m_1)\}$ are statistically-close.

The Cryptosystem. We provide the underlying DDH-based encryption scheme from [PVW08], which is a variant of the ElGammal encryption scheme:

- $\mathsf{KeyGen}(1^\lambda)$: Choose $\mathbb{G} = (G, p, q) \leftarrow \mathcal{G}(1^\lambda)$. The message space of the system is G. Choose another generator $h \leftarrow G$ and exponent $x \leftarrow \mathbb{Z}_q$. Let $pk = (g, h, g^x, h^x)$ and $sk = x$. Output (pk, sk).
- $\mathsf{Enc}(pk, m)$. Parse $pk = (g, h, g', h')$. Choose random $s, t \leftarrow \mathbb{Z}_q$. Let $u = g^s h^t$, and $v = (g')^s (h')^t$. Output $(u, v \cdot (g')^m)$.
- $\mathsf{Dec}(sk, c)$: Parse c as (c_0, c_1). Output $m \in \{0, 1\}$ for which $(g')^m = c_1 / c_0^{sk}$.

The distribution over the CRS, $\mathcal{C}$. We describe the distribution of sampling CRS for the oblivious transfer. Peikert et al. [PVW08] shows the following: There exist two different distributions $\mathcal{C}_{\mathsf{mes}}$ and $\mathcal{C}_{\mathsf{dec}}$ that are computationally-indistinguishable assuming DDH. Moreover, if the CRS is sampled according to $\mathcal{C}_{\mathsf{mes}}$ then the protocol achieves statistical security for the sender and computational security for the receiver, whereas if $\mathcal{C}_{\mathsf{dec}}$ then the protocol achieves computational security for the sender and statistical security for the receiver. We describe the two distributions:

1. $\mathcal{C}_{\mathsf{mes}}$: Choose $\mathbb{G} = (G, p, q) \leftarrow \mathcal{G}(1^\lambda)$. Choose random generators $g_0, g_1 \in G$, and choose distinct non-zero exponents $x_0, x_1 \leftarrow \mathbb{Z}_p \setminus \{0\}$. Let $h_b = g_b^{x_b}$ for $b \in \{0, 1\}$. Output $\mathsf{CRS} = (\mathbb{G}, g_0, h_0, g_1, h_1)$.

2. $\mathcal{C}_{\mathsf{dec}}$: Choose $\mathbb{G} = (G, p, q) \leftarrow \mathcal{G}(1^\lambda)$. Choose a random generator $g_0 \in G$, a random non-zero $y \leftarrow \mathbb{Z}_p \setminus \{0\}$ and let $g_1 = g_0^y$. Choose a random non-zero $x \leftarrow \mathbb{Z}_p \setminus \{0\}$. Let $h_b = g_b^x$ for $b \in \{0,1\}$, and output $\mathsf{CRS} = (\mathbb{G}, g_0, h_0, g_1, h_1)$.

The construction works with both distributions, while if $\mathcal{C}_{\mathsf{mes}}$ is used then the protocol achieves statistical security for the sender and computational security for the receiver, and if $\mathcal{C}_{\mathsf{dec}}$ is used then the security guarantee is reversed. The existence of these two distribution is important for proving the security of the protocol.

In [PVW08], it is shown that CRS can be sampled from $\mathcal{C}_{\mathsf{mes}}$ along with a trapdoor that enables extracting the input of the receiver efficiently, and this is used to prove the case of a corrupted receiver. Similarly, CRS can be sampled from $\mathcal{C}_{\mathsf{dec}}$ along with a trapdoor that enables extracting the input of the sender efficiently, and this is sued to prove the case of a corrupted sender. We describe the protocol using distribution $\mathcal{C}$ for the CRS, where each one of the $\mathcal{C}_{\mathsf{dec}}$ and $\mathcal{C}_{\mathsf{mes}}$ can be used.

The protocol. We are now ready to describe the 2-round OT protocol of [PVW08]:

Protocol 6.3: The PVW Protocol in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model

- **Input:** The sender holds bits $m_0, m_1 \in \{0,1\}$ and the receiver holds $\sigma \in \{0,1\}$.
- **The protocol:**
  1. $\mathsf{GenCRS}(1^\lambda)$: Sample $\mathsf{CRS} = (\mathbb{G}, g_0, h_0, g_1, h_1) \leftarrow \mathcal{C}$.
  2. **The receiver ($\mathbf{OT_1(CRS, \sigma)}$):** Choose a random $r \leftarrow \mathbb{Z}_p \setminus \{0\}$. Let $g = g_\sigma^r$, $h = h_\sigma^r$, and send $pk = (g, h)$ to the sender and store the secret state $sk = r$.
  3. **The sender ($\mathbf{OT_2(m_0, m_1, pk)}$):** Define $pk_0 = (g_0, h_0, g, h)$ and $pk_1 = (g_1, h_1, g, h)$. Send $c_0 = \mathsf{Enc}(pk_0, m_0)$ and $c_1 = \mathsf{Enc}(pk_1, m_1)$ to the receiver.
  4. **The receiver ($\mathbf{Output(c_0, c_1, sk)}$):** Note that $pk_\sigma = (g_\sigma, h_\sigma, g_\sigma^r, h_\sigma^r)$, and thus c_σ corresponds to encryption with $sk = r$. Output $\mathsf{Dec}(sk, c_\sigma)$.

String OT. We also consider a variant of string OT, in which the inputs of the sender are not bits but rather ℓ-bit strings. Let $\mathbf{m}_0 = (m_0[1], \ldots, m_0[\ell]), \mathbf{m}_1 = (m_1[1], \ldots, m_1[\ell])$ be the input of the sender, and let $\sigma \in \{0,1\}$ be the choice bit of the receiver. To implement string OT, the sender simply sends many encryptions $c_0 = \mathsf{Enc}(pk_0, m_0[1]), \ldots, \mathsf{Enc}(pk_0, m_0[\ell])$, and $c_1 = \mathsf{Enc}(pk_1, m_1[1]), \ldots, \mathsf{Enc}(pk_1, m_1[\ell])$. The receiver can only decrypt $\mathbf{m}_\sigma$.

6.3.1 Rerandomizability of PVW OT Protocol

We claim that the PVW protocol is rerandomizable. To argue this, we first set up some notation.

For a protocol in which the sender is run with input (m_0, m_1) and randomness (s_0, t_0, s_1, t_1), and the receiver is run with input σ and randomness r, we let $\mathsf{OT}(\mathsf{CRS}, (\sigma; r), ((m_0, m_1); (s_0, t_0, s_1, t_1)))$ denote the transcript of that execution. The transcript can be described as follows:

$$\mathsf{OT}(\mathsf{CRS}, (\sigma; r), ((m_0, m_1); (s_0, t_0, s_1, t_1))) = \begin{cases} \mathsf{CRS} = (\mathbb{G}, g_0, h_0, g_1, h_1) \\ pk = (g_\sigma^r, h_\sigma^r) \\ (c_0^0, \quad c_0^1) = (g_0^{s_0} \cdot h_0^{t_0}, \quad (g_\sigma^r)^{m_0+s_0} \cdot (h_\sigma^r)^{t_0}) \\ (c_1^0, \quad c_1^1) = (g_1^{s_1} \cdot h_1^{t_1}, \quad (g_\sigma^r)^{m_1+s_1} \cdot (h_\sigma^r)^{t_1}) \end{cases}$$

We first describe a helper rerandomization procedure that will make it easier to describe the rerandomizability algorithms associated with the PVW OT protocol.

$\mathsf{Rerand}_1(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R; R)$: Given message $(g, h) = \mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, (\sigma; r))$, we can rerandomize by simply choosing R uniformly at random in $\mathbb{Z}_p$ and output (g^R, h^R). Clearly, this is equivalent to the output of $\mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, (\sigma; rR))$. Therefore, for every r there exists a bijection $g_R(r) = rR$.

$\mathsf{Rerand}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R, \mathsf{msg}_S; S_0, T_0, S_1, T_1)$: Given a message

$$\mathsf{msg}_S = ((c_0^0, c_0^1), (c_1^0, c_1^1)) = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, m_0, m_1, \mathsf{msg}_R) \ ,$$

corresponding to some message $\mathsf{msg}_R = pk = (g, h)$, we rerandomize msg_S by simply choosing S_0, T_0, S_1, T_1 uniformly at random from $\mathbb{Z}_p$ and define:

$$\begin{aligned} (C_0^0, C_0^1) &:= \left(c_0^0 \cdot g_0^{S_0} h_0^{T_0},\ (c_0^1)^R \cdot g^{S_0} h^{T_0}\right) \ , \\ (C_1^0, C_1^1) &:= \left(c_1^0 \cdot g_1^{S_1} h_1^{T_1},\ (c_1^1)^R \cdot g^{S_1} h^{T_1}\right) \ . \end{aligned}$$

Output $((C_0^0, C_0^1), (C_1^0, C_1^1))$.

It is easy to see that for every msg_R (where $\mathsf{CRS}_{\mathsf{OT}}$ is omitted for clarity):

$$\mathsf{Rerand}_2\left(\mathsf{msg}_R, \mathsf{OT}_2\left(m_0, m_1, \mathsf{msg}_R;\ s\right); S\right) = \mathsf{OT}_2\left(m_0, m_1, \mathsf{msg}_R;\ s + S\right) \ .$$

where $s = (s_0, t_0, s_1, t_1)$ and $S = (S_0, T_0, S_1, T_1)$. Thus, we define the bijection $g_S(s) = s + S$.

$\mathsf{Rerand}_{\mathsf{OT}}(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R, \mathsf{msg}_S; R, S_0, T_0, S_1, T_1)$: Given a message

$$\mathsf{msg}_S = ((c_0^0, c_0^1), (c_1^0, c_1^1)) = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, m_0, m_1, \mathsf{msg}_R) \ ,$$

corresponding to some message $\mathsf{msg}_R = pk = (g, h)$, we first rerandomize msg_R by choosing a random $R \leftarrow \mathbb{Z}_p$ and setting: $\mathsf{msg}'_R = \mathsf{Rerand}_1(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R; R)$. Then, we rerandomize msg_S by simply choosing S_0, T_0, S_1, T_1 uniformly at random from $\mathbb{Z}_p$ and define:

$$\begin{aligned} (C_0^0, C_0^1) &:= \left(c_0^0 \cdot g_0^{S_0} h_0^{T_0},\ (c_0^1)^R \cdot g^{S_0} h^{T_0}\right) \ , \\ (C_1^0, C_1^1) &:= \left(c_1^0 \cdot g_1^{S_1} h_1^{T_1},\ (c_1^1)^R \cdot g^{S_1} h^{T_1}\right) \ . \end{aligned}$$

Output $((C_0^0, C_0^1), (C_1^0, C_1^1))$.

It is easy to see that for every msg_R (where $\mathsf{CRS}_{\mathsf{OT}}$ is omitted for clarity):

$$\begin{aligned} &\mathsf{Rerand}_2\left(\mathsf{msg}_R, \mathsf{OT}_2\left(m_0, m_1, \mathsf{msg}_R;\ s\right); R, S\right) \\ &\qquad = \mathsf{OT}_2\left(m_0, m_1, \mathsf{Rerand}_1(\mathsf{msg}_R; R);\ s + S\right) \ . \end{aligned}$$

where $s = (s_0, t_0, s_1, t_1)$ and $S = (S_0, T_0, S_1, T_1)$. Letting $\mathsf{msg}_R = \mathsf{OT}_1(\sigma, r)$ and thus $\mathsf{trans} = \mathsf{OT}((\sigma; r), (m_0, m_1; s))$,

$$\mathsf{Rerand}_{\mathsf{OT}}\left(\mathsf{trans}; R, S\right) = \mathsf{OT}\left((\sigma; rR), (m_0, m_1; s + S)\right) \ .$$

Thus, we define the bijection $g_S(s) = s + S$ and $g_R(r) = rR$.

6.4 Malicious Authority Security

While rerandomizable OT protocol will be used to construct secure two-party computation with malicious authority, as a warmup, we first construct an OT protocol with malicious authority. The ideas developed here will be useful in the secure two-party computation setting.

Tools. In the construction of two-round OT with malicious authority security, we use the following building blocks:

- A maliciously secure, two-round, rerandomizable $n \times \mathsf{OT}$ protocol in the CRS model, as in Definition 6.2. Denote the construction as OT.
- A one-way function f.
- A non-interactive witness indistinguishability proof (Definition 3.2), denoted as $\Pi^{(R)}_{\mathsf{NIWI}}$ and $\Pi^{(S)}_{\mathsf{NIWI}}$. The languages are described as part of the GenCRS algorithm of the construction.

Construction 6.4: Malicious Authority Security for Oblivious Transfer

---

$\mathsf{GenCRS}(1^\lambda)$:

1. Sample $\mathsf{CRS}_{\mathsf{OT}} \leftarrow \mathsf{OT}.\mathsf{GenCRS}(1^\lambda)$
2. **Information for corrupted sender:**
   (a) For both $b = 0$ and $b = 1$:
      i. Choose random $\boldsymbol{\sigma}_b \in \{0,1\}^n$, and compute $\Sigma_b = f(\boldsymbol{\sigma}_b)$.
      ii. Compute $\boldsymbol{\alpha}_{b,0} = \mathsf{OT}_1(\boldsymbol{\sigma}_b \oplus \mathbf{0}; R_{b,0})$ and $\boldsymbol{\alpha}_{b,1} = \mathsf{OT}_1(\boldsymbol{\sigma}_b \oplus \mathbf{1}; R_{b,1})$ for random $R_{b,0}, R_{b,1}$. Let $\boldsymbol{\alpha}_b = (\boldsymbol{\alpha}_{b,0}, \boldsymbol{\alpha}_{b,1})$ and $\mathbf{R}_b = (R_{b,0}, R_{b,1})$.
   (b) Prove that either the transcript for $b = 0$ was generated correctly, or $b = 1$. That is, we use $\Pi^{(S)}_{\mathsf{NIWI}}$ for the following relation:
   - **Instance:** $(\mathsf{CRS}_{\mathsf{OT}}, (\Sigma_0, \boldsymbol{\alpha}_0), (\Sigma_1, \boldsymbol{\alpha}_1))$ and **witness:** $(\boldsymbol{\sigma}_0, \mathbf{R}_0, \boldsymbol{\sigma}_1, \mathbf{R}_1)$.
   - **Such that:** one of the following conditions hold:
      i. $\Sigma_0 = f(\boldsymbol{\sigma}_0)$, $\boldsymbol{\alpha}_{0,0} = \mathsf{OT}_1(\boldsymbol{\sigma}_0 \oplus \mathbf{0}; R_{0,0})$, and $\boldsymbol{\alpha}_{0,1} = \mathsf{OT}_1(\boldsymbol{\sigma}_0 \oplus \mathbf{1}; R_{0,1})$, or,
      ii. $\Sigma_1 = f(\boldsymbol{\sigma}_1)$, $\boldsymbol{\alpha}_{1,0} = \mathsf{OT}_1(\boldsymbol{\sigma}_1 \oplus \mathbf{0}; R_{1,0})$, and $\boldsymbol{\alpha}_{1,1} = \mathsf{OT}_1(\boldsymbol{\sigma}_1 \oplus \mathbf{1}; R_{1,1})$.

   Compute $\pi^{(S)} \leftarrow \Pi^{(S)}_{\mathsf{NIWI}}.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{OT}}, (\Sigma_0, \boldsymbol{\alpha}_0), (\Sigma_1, \boldsymbol{\alpha}_1), (\boldsymbol{\sigma}_0, \mathbf{R}_0, \perp, \perp))$.
3. **Information for corrupted receiver:**
   (a) For both $b = 0$ and $b = 1$:
      i. Choose random $\mathsf{ch}_b \in \{0,1\}^n$ and compute $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch}_\mathsf{b})$.
      ii. Compute:

$$
\begin{aligned}
\boldsymbol{\beta}_{b,0,0,0} &= \mathsf{OT}(\mathbf{0}, (\mathbf{0},\ \mathsf{ch}_b \oplus \mathbf{0})), & \boldsymbol{\beta}_{b,1,0,0} &= \mathsf{OT}(\mathbf{1}, (\mathsf{ch}_b \oplus \mathbf{0},\ \mathbf{0})),\\
\boldsymbol{\beta}_{b,0,0,1} &= \mathsf{OT}(\mathbf{0}, (\mathbf{0},\ \mathsf{ch}_b \oplus \mathbf{1})), & \boldsymbol{\beta}_{b,1,0,1} &= \mathsf{OT}(\mathbf{1}, (\mathsf{ch}_b \oplus \mathbf{0},\ \mathbf{1})),\\
\boldsymbol{\beta}_{b,0,1,0} &= \mathsf{OT}(\mathbf{0}, (\mathbf{1},\ \mathsf{ch}_b \oplus \mathbf{0})), & \boldsymbol{\beta}_{b,1,1,0} &= \mathsf{OT}(\mathbf{1}, (\mathsf{ch}_b \oplus \mathbf{1},\ \mathbf{0})),\\
\boldsymbol{\beta}_{b,0,1,1} &= \mathsf{OT}(\mathbf{0}, (\mathbf{1},\ \mathsf{ch}_b \oplus \mathbf{1})), & \boldsymbol{\beta}_{b,1,1,1} &= \mathsf{OT}(\mathbf{1}, (\mathsf{ch}_b \oplus \mathbf{1},\ \mathbf{1}))\ . \qquad (3)
\end{aligned}
$$

      while using $\mathsf{CRS}_{\mathsf{OT}}$ in all executions, where we use randomness R'_{b,s,m_0,m_1} for the receiver and S'_{b,s,m_0,m_1} for the sender when generating the transcript $\boldsymbol{\beta}_{b,s,m_0,m_1}$, for every $b, s, m_0, m_1 \in \{0,1\}$. We let $\boldsymbol{\beta}_b := \{\boldsymbol{\beta}_{b,s,m_0,m_1}\}_{s,m_0,m_1 \in \{0,1\}}$ and define $\mathbf{S}'_b$ and $\mathbf{R}'_b$ in a similar manner.

(b) Using $\Pi^{(R)}_{\mathsf{NIWI}}$ for showing that either the transcript for $b = 0$ was generated correctly, or $b = 1$. That is, we use $\Pi^{(R)}_{\mathsf{NIWI}}$ for the following language:
- **Instance:** $(\mathsf{CRS}_{\mathsf{OT}}, (\mathsf{CHR}_0, \boldsymbol{\beta}_0, \mathbf{R}'_0)\,,\ (\mathsf{CHR}_1, \boldsymbol{\beta}_1, \mathbf{R}'_1))$.
- **Witness:** $(\mathsf{ch}_0, \mathbf{S}_0, \mathsf{ch}_1, \mathbf{S}_1)$.
- **Such that:** either for $b = 0$ or $b = 1$ it holds that $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch}_\mathsf{b})$ and $\boldsymbol{\beta}_b$ was generated as in Eq. (3) with randomness $\mathbf{R}'_b$ and $\mathbf{S}_b$.
Let $\pi^{(R)} \leftarrow \Pi^{(R)}_{\mathsf{NIWI}}.\mathsf{Prove}\left((\mathsf{CRS}_{\mathsf{OT}}, (\mathsf{CHR}_0, \boldsymbol{\beta}_0, \mathbf{R}'_0)\,,\ (\mathsf{CHR}_1, \boldsymbol{\beta}_1, \mathbf{R}'_1))\,, (\mathsf{ch}_0, \mathbf{S}_0, \bot, \bot)\right)$.

4. **Output:**

$$\mathsf{CRS} = \left(\mathsf{CRS}_{\mathsf{OT}}, (\Sigma_0, \boldsymbol{\alpha}_0, \mathbf{S}_0), (\Sigma_1, \boldsymbol{\alpha}_1, \mathbf{S}_1), (\mathsf{CHR}_0, \boldsymbol{\beta}_0, \mathbf{R}'_0), (\mathsf{CHR}_1, \boldsymbol{\beta}_1, \mathbf{R}'_1), \pi^{(\mathsf{R})}, \pi^{(\mathsf{S})}\right)\ . \quad (4)$$

The implementation of the OT protocol is the same as the base one, where upon receiving CRS we just take the first coordinate $\mathsf{CRS}_{\mathsf{OT}}$.

OT: The OT protocol is the same as the underlying OT, where upon receiving $\mathsf{CRS} = (\mathsf{CRS}_{\mathsf{OT}}, \ldots)$ the algorithm only takes $\mathsf{CRS}_{\mathsf{OT}}$.

Judge(CRS, τ).

1. Parse: CRS as in Eq. (4). Run $\Pi^{(R)}_{\mathsf{NIWI}}.\mathsf{Verify}$ to verify $\pi^{(R)}$ on the CRS was generated correctly, and run $\Pi^{(S)}_{\mathsf{NIWI}}.\mathsf{Verify}$ to verify that $\pi^{(S)}$ was generated correctly. If the verification did not pass, then output $\mathsf{corrupted}$.
2. Otherwise, there are two options:
   (a) If $\tau = (b, \mathsf{sender}, \boldsymbol{\sigma}_b)$, then verify that $\Sigma_b = f(\boldsymbol{\sigma}_b)$. If so, output $\mathsf{corrupted}$.
   (b) If $\tau = (b, \mathsf{receiver}, \mathsf{ch}_b)$, then verify that $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch}_\mathsf{b})$. If so, output $\mathsf{corrupted}$.

Theorem 6.5. *Construction 6.4 has malicious authority security for to the functionality $n \times \mathsf{OT}$ with respect to the uniform distribution over the inputs.*

Proof. According to the definition of malicious authority security (Definition 4.2) we have to show simulatability of the OT protocol, accountability and defamation free.

Simulation. The construction is identical to that of [PVW08], where we just change the CRS and add some additional information to it.

Accountability: sender. Fix an adversary $\mathcal{A}$, and assume that the adversary corrupts the sender. We show that if there exists a non-negligible function $\epsilon_1(\cdot)$ for which

$$\Pr[\mathsf{Acc.Real}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda) = 1] \geq \epsilon_1(\lambda)\ ,$$

then there exists a non-negligible function $\epsilon_2(\cdot)$ such that

$$\Pr[\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda) = 1] \geq \epsilon_2(\lambda)\ .$$

Recall that the adversary $\mathcal{A}$ outputs a common reference string CRS^* and then the protocol is executed for q number of times, where the adversary chooses the input for the sender, the input of the receiver is chosen according to the uniform distribution and the adversary receives the resulting transcript. The adversary $\mathcal{A}$ then wins if it succeeds to output the input of the sender in one of the q executions.

The extractor Ext works as follows:

1. Invoke the adversary $\mathcal{A}$ and obtain CRS^*.
2. Check that CRS^* is well formed. I.e., run Step 1 in the Judge algorithm. If the CRS^* is corrupted, then output $\tau = \mathsf{CRS}^*$ and halt.
3. Choose a random $b \leftarrow \{0,1\}$, and run the following for q iterations where in the kth execution:
   (a) Sample a masking shift $\boldsymbol{\Delta}^{(k)} = (\Delta_1, \ldots, \Delta_n)$ uniformly at random for the input of the receiver. The procedure will generate a transcript where the input of the receiver is $\boldsymbol{\sigma}_b \oplus \boldsymbol{\Delta}^{(k)}$ (with some fresh uniform random tape).
   (b) Receive from the adversary $\mathcal{A}$ the input $(\mathbf{m}_0, \mathbf{m}_1)$ and randomness $\boldsymbol{\rho}$.
   (c) For every $i \in [n]$ set $\mathsf{msg}_R[i]$ as follows:
      i. The goal is to obtain a bit-OT for input $\boldsymbol{\sigma}_b[i] \oplus \Delta_i$ for the receiver and $(\mathbf{m}_0[i], \mathbf{m}_1[i])$ for the sender, with randomness $\boldsymbol{\rho}[i]$ for the sender.
      ii. $\alpha = \alpha_{b,\Delta_i}[i]$ corresponds to 1-out-of-2 bit OT with receiver using input $\boldsymbol{\sigma}_b[i] \oplus \Delta_i$. Set $\mathsf{msg}_R[i] \leftarrow \mathsf{OT.Rerand}_1(\alpha)$ for a uniform random tape.
   (d) Run $\mathsf{msg}_S = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R, \mathbf{m}_0, \mathbf{m}_1, \boldsymbol{\rho})$, and set $\mathsf{trans}_k = (\mathbf{m}_0, \mathbf{m}_1, \boldsymbol{\rho}, \mathsf{msg}_R, \mathsf{msg}_S)$. Send trans_k to the adversary $\mathcal{A}$.
4. Let $(k, \mathbf{y}^{(k)})$ be the output of the adversary $\mathcal{A}$, where the input $\mathbf{y}^k$ represents the input of the receiver in the kth execution. Set $\boldsymbol{\sigma}'_b = \boldsymbol{\Delta}^{(k)} \oplus \mathbf{y}^{(k)}$.
5. If $f(\boldsymbol{\sigma}'_b) = \Sigma_b$ then set $\tau = (b, \mathsf{sender}, \boldsymbol{\sigma}'_b)$ and output it. Otherwise, output fail.

If the malicious authority outputs CRS^* that does not pass the verification, then the output of $\mathsf{Acc.Ext}$ is 1. We therefore assume that this happens with negligible probability.

We now prove that the view of the adversary is identical in both executions, condition that the extractor choose the correct branch b. Since the verification of the CRS^* is succeeded, then it must be that the malicious authority used a correct witness in computing $\pi^{(R)}$ (either $(\boldsymbol{\sigma}_0, \mathbf{R}_0, \perp, \perp)$ or $(\perp, \perp, \boldsymbol{\sigma}_1, \mathbf{R}_1)$), and we condition on b accordingly. To show that, we just take a closer look at the real and ideal executions:

1. $\mathsf{Acc.Real}$: In the real execution, in each iteration the malicious authority gives some $(\mathbf{m}_0, \mathbf{m}_1)$ and randomness $\boldsymbol{\rho}$. The honest reciever chooses its input $\mathbf{r}$ according to the uniform distribution, and the malicious authority receives $\mathsf{OT}(\mathbf{r}, (\mathbf{m}_0, \mathbf{m}_1; \boldsymbol{\rho}))$.
2. $\mathsf{Acc.Ext}$: In the ideal execution, in each iteration the malicious authority gives some $(\mathbf{m}_0, \mathbf{m}_1)$ and randomness $\boldsymbol{\rho}$. The extractor chooses random shift $\boldsymbol{\Delta}$ according to the uniform distribution, and the malicious authority uses the CRS to create a transcript that correspond to the execution $\mathsf{OT}(\boldsymbol{\Delta} \oplus \boldsymbol{\sigma}_b, (\mathbf{m}_0, \mathbf{m}_1; \boldsymbol{\rho}))$.

In both cases, the malicious authority receives a transcript that correspond to a uniform random input of the receiver. The transcript includes the input and randomness of the sender, but from the security of the OT, this gives no information about the input of the receiver. As such, its view is identical in both executions.

From inspection, it is clear that once the extractor receives from the malicious authority the input for which it had created the transcript, it can easily recover $\boldsymbol{\sigma}_b$. This implies that if the malicious authority succeeds in the real with some non-negligible probability $\epsilon_1(\lambda)$, then the extractor succeeds with probability $\epsilon_2(\lambda) := 1/2\epsilon_1(\lambda)$, where $1/2$ comes from the guessing of b.

Accountability: receiver. Fix an adversary $\mathcal{A}$ and assume that the adversary corrupts the receiver. We describe the extractor Ext:

1. Invoke the adversary $\mathcal{A}$ and obtain CRS^*.

2. Check that CRS^* is well formed. That is, run Step 1 in the Judge algorithm. If the CRS^* is corrupted, then output $\tau = \mathsf{CRS}^*$ and halt.
3. Choose a random $b \leftarrow \{0,1\}$ and run the following for q iterations where in the kth execution:
   (a) Sample an input $\mathbf{m}^{(k)} = \mathbf{m} \in \{0,1\}^n$ and a shift $\boldsymbol{\Delta}^{(k)} = \boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_n) \in \{0,1\}^n$.
   (b) Receive from the adversary the input $\mathbf{s}^{(k)} = (s_1, \ldots, s_n)$ and randomness $\boldsymbol{\rho}_R$.
   (c) Generate transcript trans_k. For every $i \in [n]$ set $\mathsf{trans}_k[i]$ as follows:
      i. The goal is to obtain a transcript for bit-OT for input s_i for the receiver with randomness $\boldsymbol{\rho}_R[i]$, and the input of the sender is:
         A. $(\mathbf{m}[i], \mathsf{ch}_b[i] \oplus \Delta_i)$ if $s_i = 0$, and
         B. $(\mathsf{ch}_b[i] \oplus \Delta_i, \mathbf{m}[i])$ if $s_i = 1$.
         To ease notation, let $s := s[i]$, $\rho = \boldsymbol{\rho}_R[i]$, $m = \mathbf{m}[i]$, $\Delta = \Delta_i$.
      ii. Let $\mathsf{trans}[i]' = \boldsymbol{\beta}_{b,s,m,\Delta}[i]$ and $R_b := R_{b,s,m,\Delta}$.
      iii. Run $\mathsf{trans}_k[i] = \mathsf{Rerand}_{\mathsf{OT}}(\mathsf{trans}[i]'; \rho', S)$ using a uniform randomness S for the sender, and for some ρ' for which $g_{\rho'}(R_b) = \rho$.
   Send trans_k to the adversary.
4. Let $(k, x^{(k)})$ be the output of the adversary $\mathcal{A}$, where $x^{(k)}$ is some pair $\mathbf{y}_0 = (y_0[1], \ldots, y_0[n])$ and $\mathbf{y}_1 = (y_1[1], \ldots, y_n[1])$. Let $\mathbf{m} = \mathbf{m}^{(k)}$, $\boldsymbol{\Delta} := \boldsymbol{\Delta}^{(k)}$ and $\mathbf{s} = \mathbf{s}^{(k)}$, as in Steps 3a and 3b in the kth execution. Set ch' where for every $i \in [n]$:

$$\mathsf{ch}'[i] = \begin{cases} y_1[i] \oplus \Delta_i & \text{If } s_i = 0 \\ y_0[i] \oplus \Delta_i & \text{otherwise} \end{cases}$$

5. If $f(\mathsf{ch}') = \mathsf{CHR}_\mathsf{b}$, then set $\tau = (b, \mathsf{receiver}, \mathsf{ch}')$ and output it. Otherwise, output fail.

As in the case of the sender, the view of the adversary is distributed identically between the real and ideal, conditioned on the event that the extractor guessed b correctly. In both execution the input and randomness of the receiver are $\mathbf{s}, \boldsymbol{\rho}_R$, respectively. In the real execution the input of the sender corresponds to uniformly random messages $\mathbf{m}_0, \mathbf{m}_1$. In the extraction, the input of the sender is also uniform.

It is easy to see that once the malicious authority outputs an input used in one of the transcripts, then the extractor learns the secret $\mathbf{x}$ in the CRS^*. Therefore, if the malicious authority succeeds with some probability $\epsilon_1(\lambda)$ in the real execution then the extractor succeeds with probability $\epsilon_2(\lambda) := 1/2 \cdot \epsilon_1(\lambda)$.

Defamation free. We show that for every non-uniform adversary $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for all λ:

$$\Pr\left[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}\right] \leq \mu(\lambda) \ ,$$

where $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$.

Fix an adversary $\mathcal{A}$. Since CRS is honestly generated, the check in Step 1 always passes. We thus can write:

$$\begin{aligned} \Pr\left[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}\right] \quad = \quad & \Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{sender}, \boldsymbol{\sigma}) \text{ s.t. } \Sigma_b = f(\boldsymbol{\sigma})\right] \\ & + \Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})\right] \end{aligned}$$

The following two claims (Claim 6.6 and 6.7) show show that the probability of each one of these terms is negligible, concluding the proof.

Claim 6.6. *For every $b \in \{0,1\}$ there exists a negligible function $\mu_b(\cdot)$ such that:*

$$\Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{sender}, \boldsymbol{\sigma}) \text{ s.t. } \Sigma_b = f(\boldsymbol{\sigma})\right] \leq \mu_b(\lambda) \ .$$

Proof. Fix $b \in \{0,1\}$. We prove the claim via a sequence of hybrid experiments. Specifically:

1. $\mathbf{H}_1$: in this experiment, the adversary receives CRS as generated by GenCRS, and the output of the experiment is 1 if it outputs $(b, \mathsf{sender}, \boldsymbol{\sigma})$ such that $\Sigma_b = f(\boldsymbol{\sigma})$.
2. $\mathbf{H}_2$: in this experiment, we change the CRS that the adversary receives. In $\pi^{(R)}$ we use the witness $(\bot, \bot, \boldsymbol{\sigma}_1, \mathbf{R}_1)$ if $b = 0$, and use the witness $(\boldsymbol{\sigma}_0, \mathbf{R}_0, \bot, \bot)$ if $b = 1$ (i.e., we use the other branch). Again, the adversary receives the generated CRS and the output of the experiment is 1 if it outputs $(b, \mathsf{sender}, \boldsymbol{\sigma})$ such that $\Sigma_b = f(\boldsymbol{\sigma}_b)$.
3. $\mathbf{H}_3$: This is the same as the previous experiment, but for which we change $\boldsymbol{\alpha}_{b,0} = \mathsf{OT}_1(\mathbf{0}; R_{b,0})$ instead of $\mathsf{OT}_1(\boldsymbol{\sigma}_b \oplus \mathbf{0})$.
4. $\mathbf{H}_4$: This is the same as the previous experiment, but for which we change $\boldsymbol{\alpha}_{b,1} = \mathsf{OT}_1(\mathbf{1}; R_{b,1})$ instead of $\mathsf{OT}_1(\boldsymbol{\sigma}_b \oplus \mathbf{1})$.

We now claim that the view of the adversary is computationally-indistinguishable between any two consecutive hybrids, which implies that the probability that the adversary outputs $\boldsymbol{\sigma}$ such that $\Sigma_b = f(\boldsymbol{\sigma})$ is negligibly-close between two consecutive hybrids.

- The view of the adversary in $\mathbf{H}_1$ and $\mathbf{H}_2$ is computationally-indistinguishable based on the security of the NIWI.
- The view of the adversary between $\mathbf{H}_2$ and $\mathbf{H}_3$ is computationally indistinguishability from the security of the OT protocol: The CRS is different in only the way $\boldsymbol{\alpha}_{b,0}$ is generated: In $\mathbf{H}_2$ it is generated as $\mathsf{OT}_1(\boldsymbol{\sigma}_b \oplus \mathbf{0})$ whereas in $\mathbf{H}_3$ it is generated as $\mathsf{OT}_1(\mathbf{0})$. In both cases, the randomness of the receiver is not known to the adversary. From the security of OT, the view of the sender is indistinguishable in both executions. Moreover, recall that the NIWI is proven using the witness $(1-b, \boldsymbol{\sigma}_{1-b}, \mathbf{R}_b)$ so this change does not affect the validity of the $\pi^{(R)}$.
- We repeat the same argument as before for $\mathbf{H}_4$.

Finally, we claim that the probability that the adversary outputs $(b, \mathsf{sender}, \boldsymbol{\sigma}_b)$ in $\mathbf{H}_4$ is negligible. The only information that the adversary receives about $\boldsymbol{\sigma}_b$ is Σ_b, as all other parts in the CRS are completely independent of $\boldsymbol{\sigma}_b$. Since $\boldsymbol{\sigma}_b$ is chosen uniformly at random by GenCRS, this is reduced to the security of the one-way function f. □

Claim 6.7. *For every $b \in \{0,1\}$ there exists a negligible function $\mu_b(\cdot)$ such that:*

$$\Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})\right] \leq \mu_b(\lambda) \ .$$

Proof. Fix $b \in \{0,1\}$. We prove the claim via hybrid argument.

1. $\mathbf{H}_1$: in this experiment, the adversary receives CRS as generated by GenCRS and the output of the experiment is 1 if it outputs $(b, \mathsf{receiver}, \mathsf{ch})$ such that $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})$.
2. $\mathbf{H}_2$: in this experiment, we change the CRS that the adversary receives. In $\pi^{(R)}$ we use the witness $(\bot, \bot, \mathsf{ch}_1, \mathbf{S}_1)$ if $b = 0$, and witness $(\mathsf{ch}_0, \mathbf{S}_0, \bot, \bot)$ if $b = 1$. Again, the adversary receives the generated CRS and the output of the experiment is 1 if it outputs $(b, \mathsf{receiver}, \mathsf{ch})$ such that $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})$.

3. $\mathbf{H}_3$: this is like the previous experiment, but for which for which we remove ch from $\boldsymbol{\beta}_{b,0,\cdot,\cdot}$. That is, set:

$$\boldsymbol{\beta}_{b,0,0,0} = \mathsf{OT}(\mathbf{0},(\mathbf{0},\mathbf{0})), \qquad \boldsymbol{\beta}_{b,0,0,1} = \mathsf{OT}(\mathbf{0},(\mathbf{0},\mathbf{1}))$$
$$\boldsymbol{\beta}_{b,0,1,0} = \mathsf{OT}(\mathbf{0},(\mathbf{1},\mathbf{0})), \qquad \boldsymbol{\beta}_{b,0,1,1} = \mathsf{OT}(\mathbf{0},(\mathbf{1},\mathbf{1}))$$

4. $\mathbf{H}_4$: as the previous experiment, but we also remove ch from $\boldsymbol{\beta}_{b,1,\cdot,\cdot}$. That is, set:

$$\boldsymbol{\beta}_{b,1,0,0} = \mathsf{OT}(\mathbf{0},(\mathbf{0},\mathbf{0})), \qquad \boldsymbol{\beta}_{b,1,0,1} = \mathsf{OT}(\mathbf{0},(\mathbf{0},\mathbf{1}))$$
$$\boldsymbol{\beta}_{b,1,1,0} = \mathsf{OT}(\mathbf{0},(\mathbf{1},\mathbf{0})), \qquad \boldsymbol{\beta}_{b,1,1,1} = \mathsf{OT}(\mathbf{0},(\mathbf{1},\mathbf{1}))$$

Similar to the pervious proof,

- The view of the adversary in $\mathbf{H}_1$ and $\mathbf{H}_2$ is computationally indistinguishable directly from the security of the NIWI.
- The view of the adversary in $\mathbf{H}_2$ and $\mathbf{H}_3$ is computationally indistinguishable from the security of the OT protocol. Specifically, the transcript correspond to the view of the receiver in this execution, and includes the input and randomness of the receiver, and the messages it receivers from the sender. From the security of OT, when the input of the receiver is $\mathbf{0}$ it gets no information about the other input of the sender, i.e., it cannot distinguish whether the sender is using $\mathsf{ch}_b \oplus \mathbf{m}$ or $\mathbf{m}$.
- The view of the adversary in $\mathbf{H}_3$ and $\mathbf{H}_4$ is computationally indistinguishable based again on the security of OT in a similar manner.

Finally, we claim that the probability that the adversary outputs $(b, \mathsf{receiver}, \boldsymbol{\sigma}_b)$ in $\mathbf{H}_4$ is negligible. The only information that the adversary receives about ch_b is $f(\mathsf{ch}_b)$, as all other parts in the CRS are completely independent of ch_b. Since ch_b is chosen uniformly at random by GenCRS, this is reduced to the security of the one-way function f. Since the view of the adversary in $\mathbf{H}_1$ is computationally indistinguishable from its view in $\mathbf{H}_4$, we conclude that the probability that the adversary wins in $\mathbf{H}_1$ is negligible as well. □

□

6.5 Oblivious Transfer with Strong Accountability

The protocol described in the previous section satisfies Definition 4.2, but it does not satisfy strong accountability (Definition 4.3).

In fact, Construction 6.4 already satisfies the stronger notion for the case of a corrupted sender, but not for the corrupted receiver. We show now that using indistinguishability obfuscation, we can achieve better and achieve the stronger notion of accountability in which the malicious authority is active during the protocol execution.

The idea is that given message msg_R sent by the malicious receiver, the extractor can generate msg_S that corresponds to an execution while using messages $\mathbf{m}_0, \mathbf{m}_1$ in which the challenge ch_b is embedded in position that do not effect the output. This is done using an obfuscated circuit that is given in the CRS.

Construction 6.8: Strong Accountability for Oblivious Transfer

$\mathsf{GenCRS}(1^\lambda)$:

1. Sample $\mathsf{CRS}_{\mathsf{OT}} \leftarrow \mathsf{OT}.\mathsf{GenCRS}(1^\lambda)$.

2. **Information for corrupted sender:** Identical to Construction 6.4. Let $(\Sigma_0, \boldsymbol{\alpha}_0, \mathbf{S}_0)$, $(\Sigma_1, \boldsymbol{\alpha}_1, \mathbf{S}_1)$, $\pi^{(S)}$ be the output of this part.
3. **Information for the corrupted receiver:**
   (a) For both $b = 0$ and $b = 1$:
      i. Choose random $\mathsf{ch}_b \in \{0,1\}^n$ and compute $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch_b})$.
      ii. Choose a random PRF key $K_b \leftarrow \{0,1\}^\lambda$, randomness r_b and obfuscate Circuit 6.9:

$$G_b = \mathsf{iO.Obf}\left(G_{b,\mathsf{OT}_2}[\mathsf{CRS_{OT}}, \mathsf{ch}_b, K_b]; \mathsf{r}_b\right) \ .$$

   (b) Use $\Pi^{(R)}_{\mathsf{NIWI}}$ for showing that either the transcript for $b = 0$ was generated correctly, or $b = 1$. That is, use $\Pi^{(R)}_{\mathsf{NIWI}}$ for the following language:

$$\textbf{Instance}: (\mathsf{CRS_{OT}}, (X_0, G_0), (X_1 G_1)) \qquad \textbf{Witness}: ((\mathsf{ch}_0, K_0, \mathsf{r}_0), (\mathsf{ch}_1, K_1, \mathsf{r}_1)) \ ,$$

   such that for either $b = 0$ or $b = 1$ it holds that:

$$\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch_b}) \qquad \text{and} \qquad \mathsf{G_b} = \mathsf{iO.Obf}\left(\mathsf{G}_{\mathsf{b},\mathsf{OT}_2}[\mathsf{CRS_{OT}}, \mathsf{ch_b}, \mathsf{K_b}]; \mathsf{r_b}\right)$$

   Let $\pi^{(R)} = \Pi^{(R)}_{\mathsf{NIWI}}.\mathsf{Prove}\Big(\mathsf{CRS_{OT}}, (X_0, G_0), (X_1, G_1), ((\mathsf{ch}_0, K_0, \mathsf{r}_0), (\bot, \bot, \bot))\Big)$.
4. Output: $\left(\mathsf{CRS_{OT}}, (\Sigma_0, \boldsymbol{\alpha}_0, \mathbf{S}_0), (\Sigma_1, \boldsymbol{\alpha}_1, \mathbf{S}_1), (X_0, G_0), (X_1 G_1), \pi^{(S)}, \pi^{(R)}\right)$.

The protocol is identical to that of Construction 6.4.

The Judge algorithm are identical to Construction 6.4, where the only difference is the underlying language of $\Pi^{(R)}_{\mathsf{NIWI}}$.

Circuit 6.9: $G_{b,\mathsf{OT}_2}[\mathsf{CRS_{OT}}, \mathsf{ch}_b, K](\mathsf{msg}_R)$**:**

- **Input:** msg_R.
- **Hardwired value:** $\mathsf{CRS_{OT}}$, ch_b, a pseudorandom key K.
- **The circuit:**
  1. Let $\left(\mathbf{m}_0, \mathbf{m}_1, \{\mathsf{r}^{\tau_0,\tau_1}_{\mathsf{OT}_2}\}_{\tau_0,\tau_1\in\{0,1\}}\right) \leftarrow \mathsf{PRF}_K\left(\mathsf{msg}_R\right)$.
  2. For every $\tau_1, \tau_2 \in \{0,1\}$, compute $\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS_{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \mathbf{m}_1 \oplus \tau_1^n, \mathsf{msg}_R; \mathsf{r}^{\tau_0,\tau_1}_{\mathsf{OT}_2})$.
  3. **Output:** $(\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathsf{ch}_b, \{\mathsf{msg}_S\}_{\tau_0,\tau_1})$.

Theorem 6.10. *Assuming the security of one-way function, indistinguishability obfuscator, non-interactive witness-indistinguishability proof system, and the existence of a rerandomizable oblivious transfer, Construction 6.4 has malicious authority security for to the functionality $n\times\mathsf{OT}$ with respect to the uniform distribution over the inputs.*

Proof. Simulation follows since the protocol is identical to that of [PVW08] where we just add more information to the CRS.

Strong accountability: Sender. The extractor is identical to that in Theorem 6.5, except that it does not receive from $\mathcal{A}$ the input $(\mathbf{m}_0, \mathbf{m}_1)$ and randomness $\boldsymbol{\rho}$ from $\mathcal{A}$ in Step 3b. Moreover, after generating msg_R, it does not compute msg_S as described in Step 3d and instead just sends it to the malicious authority $\mathcal{A}$. The view of the adversary is identically distributed to the one in the real execution, and the extractability is also obtained from similar reasoning.

Strong accountability: Receiver. We define the extractor Ext:

1. Receive from the malicious authority CRS^*.
2. Pick a branch $b \leftarrow \{0,1\}$.
3. For each iteration $k \in [q]$:
   (a) Receive from the adversary $\mathsf{msg}_R^{(k)}$.
   (b) Compute $\mathsf{iO.Eval}(G_b, \mathsf{msg}_R^{(k)})$ to obtain $(\mathbf{y}^{(k)}, \{\mathsf{msg}_S^{\tau_0,\tau_1}\}_{\tau_0,\tau_1})$ where $\mathbf{y}^{(k)} = \mathbf{m}_0^{(k)} \oplus \mathbf{m}_1^{(k)} \oplus \mathsf{ch}_b$).
   (c) Select messages $\hat{\mathbf{m}}_0^{(k)} = (\hat{m}_1^0, \ldots, \hat{m}_n^0)$ and $\hat{\mathbf{m}}_1^{(k)} = (\hat{m}_1^1, \ldots, \hat{m}_n^1)$ uniformly at random. The extractor will generate a transcript that correspond to using the input $(\mathbf{m}_0^{(k)} \oplus \hat{\mathbf{m}}_0^{(k)},\ \mathbf{m}_1^{(k)} \oplus \hat{\mathbf{m}}_1^{(k)})$.
   (d) For every $i \in [n]$, $\mathsf{msg}_S[i]' = \mathsf{msg}_S^{\hat{m}_i^0, \hat{m}_i^1}$.
   (e) Run $\mathsf{msg}'_S \leftarrow \mathsf{Rerand}_2(\mathsf{msg}_R, \mathsf{msg}'_S; S)$, for a uniform S.
4. If the malicious authority gives back some $(k, \mathbf{m}_0, \mathbf{m}_1)$, then output $(\mathsf{receiver}, b, \mathbf{y}^{(k)} \oplus \mathbf{m}_0 \oplus \hat{\mathbf{m}}_0^{(k)} \oplus \mathbf{m}_1 \oplus \hat{\mathbf{m}}_1^{(k)})$.

It is easy to see that conditioned on the correct branch b, the view of the malicious authority is computationally indistinguishable as generated by the extractor from its view when interacting with an honest sender in the real execution. This follows from the rerandomization of the message sent by the extractor, the perfect correctness of the obfuscation, and the security of the pseudorandom function (thus, the sampled message $\mathsf{msg}_0, \mathsf{msg}_1$ are indistinguishable from messages as sampled by the real sender). Moreover, if the malicious authority gives correct input $(\mathbf{m}_0, \mathbf{m}_1)$ for some iteration $k \in [q]$, then again from the perfect correctness of the obfuscator, the extractor can extract ch_b as given by the output of the obfuscated circuit. Therefore, if the malicious authority succeeds with some probability $\epsilon_1(\lambda)$ in the real execution, then the extractor succeeds with probability $1/2 \cdot \epsilon(\lambda) - \mathsf{negl}(\lambda)$, where $1/2$ comes from guessing b, and $\mathsf{negl}(\lambda)$ is the security loss of the PRF.

Defamation free. We show that for every non-uniform adversary $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for all λ:

$$\Pr\left[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}\right] \leq \mu(\lambda)\ ,$$

where $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$.

Fix an adversary $\mathcal{A}$. Since CRS is honestly generated, the check in Step 1 always passes. We thus can write:

$$\begin{aligned}\Pr\left[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}\right] \quad = \quad & \Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{sender}, \boldsymbol{\sigma}) \text{ s.t. } \Sigma_b = f(\boldsymbol{\sigma})\right] \\ & + \Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})\right]\end{aligned}$$

We show that each even occurs with a negligible probability. The case of a corrupted sender is identical to that of Theorem 6.5, and we derive it from Claim 6.6. We show the the second term is also bounded by a negligible function, and the following claim concludes our proof.

Claim 6.11. *For every $b \in \{0,1\}$ there exists a negligible function $\mu_b(\cdot)$ such that:*

$$\Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})\right] \leq \mu_b(\lambda)\ .$$

Proof. Suppose not. Then there exists a PPT adversary $\mathcal{A}$ and a bit b such that $\Pr[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})]$ is δ, where δ is a non-negligible function. Without loss of generality, we assume that $b = 0$. Consider the following hybrids.

Note: in the hybrids below, we abuse notation and interpret boolean strings as integers. For instance, the all zeroes string is interpreted as integer 0.

$\mathbf{H_0}$: In this hybrid, generate the common reference string as $\mathsf{CRS} \leftarrow \mathsf{GenCRS}(1^\lambda)$.

By definition, we have that $\Pr[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})]$ is δ.

$\mathbf{H_1}$: In this hybrid, we modify the CRS generation and instead generate the NIWI proof $\pi^{(R)}$ using the witness $((\bot, \bot, \bot), (\mathsf{ch}_1, K_1, \mathsf{r}_1))$. Moreover, we change the $\mathsf{CRS_{OT}}$ to be as in the simulation of the OT, define for corrupted receiver.

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{1}}$.

$\mathbf{H_{2.i}}$: We instead generate the common reference string CRS as follows. Puncture the key K_b at the index i; call this $K_b \setminus \{i\}$. Generate the obfuscated circuit as follows:

$$\widehat{\mathbf{H_{2.i}}.G_b} = \mathsf{iO.Obf}\left(\mathbf{H_2}.G_{b,\mathsf{OT}_2}[i, \mathsf{CRS_{OT}}, \mathsf{ch}_b, K_b \setminus \{i\}, \mathbf{y}]; \mathsf{r}_b\right) \ ,$$

where $\mathbf{H_2}.G_{b,\mathsf{OT}_2}$ is Circuit 6.12. Let $\mathbf{s}$ be the output of G_{b,OT_2} on i, where G_{b,OT_2} is defined in Circuit 6.9.

Circuit 6.12: $\mathbf{H_2}.G_{b,\mathsf{OT}_2}[i, \mathsf{CRS_{OT}}, \mathsf{ch}_b, K \setminus \{i\}, \mathbf{s}](\mathsf{msg}_R)$:

- **Input:** msg_R.
- **Hardwired value:** index i, $\mathsf{CRS_{OT}}$, ch_b, a punctured pseudorandom key $K \setminus \{i\}$ and value $\mathbf{s}$.
- **The circuit:**
  1. If $\mathsf{msg}_R > i$:
     (a) Let $\left(\mathbf{m}_0, \mathbf{m}_1, \{\mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}\}_{\tau_0,\tau_1 \in \{0,1\}}\right) \leftarrow \mathsf{PRF}_K(\mathsf{msg}_R)$.
     (b) For every $\tau_1, \tau_2 \in \{0,1\}$, compute $\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS_{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \mathbf{m}_1 \oplus \tau_1^n, \mathsf{msg}_R; \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1})$.
     (c) **Output:** $(\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathsf{ch}_b, \{\mathsf{msg}_S\}_{\tau_0,\tau_1})$.
  2. If $\mathsf{msg}_R = i$: output $\mathbf{s}$.
  3. If $\mathsf{msg}_R < i$:
     (a) Let $\left(\mathbf{m}_0, \mathbf{m}_1, \{\mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}\}_{\tau_0,\tau_1 \in \{0,1\}}\right) \leftarrow \mathsf{PRF}_K(\mathsf{msg}_R)$.
     (b) For every $\tau_1, \tau_2 \in \{0,1\}$, compute $\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS_{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \mathbf{m}_1 \oplus \tau_1^n, \mathsf{msg}_R; \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1})$.
     (c) **Output:** $(\mathbf{m}_0 \oplus \mathbf{m}_1, \{\mathsf{msg}_S\}_{\tau_0,\tau_1})$.

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{2.i}}$.

$\mathbf{H_{3.i}}$: We modify the CRS generation from the previous hybrid ($\mathbf{H_{2.i}}$) as follows. The value $\mathbf{s}$ hardwired in the circuit $\mathbf{H_2}.G_{b,\mathsf{OT}_2}$ is generated as $(\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathsf{ch}_b, \{\mathsf{msg}_S^{\tau_0,\tau_1}\}_{\tau_0,\tau_1})$, where each $\mathsf{msg}_S^{\tau_0,\tau_1}$ is generated as follows:

$$\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS_{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \mathbf{m}_1 \oplus \tau_1^n, i; \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1})$$

where $\mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}$ is sampled uniformly at random (instead of an output of the PRF).

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{3.i}}$.

$\mathbf{H_{4.i}}$: We further modify the CRS generation from the previous hybrid ($\mathbf{H_{3.i}}$) as follows. The hardwired value $\mathbf{s}$ is generated as $(\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathbf{s}, \mathsf{msg}_S)$, where $\mathsf{msg}_S^{\tau_0,\tau_1}$ is now defined as follows:

$$\mathsf{msg}_S^{\tau_0,\tau_1} = \begin{cases} \mathsf{OT}_2(\mathsf{CRS_{OT}}, \quad \mathbf{0}, \quad \mathbf{m}_1 \oplus \tau_1^n, \quad i; \quad \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}) & \text{if } \mathbf{b}^* = 1 \\ \mathsf{OT}_2(\mathsf{CRS_{OT}}, \quad \mathbf{m}_0 \oplus \tau_0^n, \quad \mathbf{0}, \quad i; \quad \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}) & \text{if } \mathbf{b}^* = 0 \end{cases},$$

where $\mathbf{b}^*$ is the extracted bit of the simulator upon receiving the message $\mathsf{msg}_R = i$ from the corrupted receiver in the simulation process.

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{4.i}}$.

$\mathbf{H_{5.i}}$: We further modify the CRS generation from the previous hybrid ($\mathbf{H_{4.i}}$) as follows. We generate the hardwired value $\mathbf{s}$ as follows: $\mathbf{s} = (\mathbf{m}_0 \oplus \mathbf{m}_1, \{\mathsf{msg}_S^{\tau_0,\tau_1}\}_{\tau_0,\tau_1})$, where $\{\mathsf{msg}_S^{\tau_0,\tau_1}\}_{\tau_0,\tau_1}$ is generated as in the previous hybrid.

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{5.i}}$.

$\mathbf{H_{6.i}}$: We further modify the CRS generation from the previous hybrid ($\mathbf{H_{5.i}}$) as follows. We generate the hardwired value $\mathbf{s}$ as follows: $\mathbf{s} = (\mathbf{m}_0 \oplus \mathbf{m}_1, \{\mathsf{msg}_S^{\tau_0,\tau_1}\}_{\tau_0,\tau_1})$, where $\mathsf{msg}_S^{\tau_0,\tau_1}$ is generated as follows for every $\tau_0, \tau_1 \in \{0,1\}$:

$$\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS_{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \ \mathbf{m}_1 \oplus \tau_1^n, \quad i; \quad \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}),$$

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{6.i}}$.

$\mathbf{H_{7.i}}$: We further modify the CRS generation from the previous hybrid ($\mathbf{H_{6.i}}$) as follows. We generate the hardwired value $\mathbf{s}$ as in the previous hybrid, where each $\mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}$ is generated to be the output of $\mathsf{PRF}_K(i)$ (instead of a uniform one).

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR_b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{7.i}}$.

$\mathbf{H_{8.i}}$: We instead further modify the generation of the common reference string CRS as follows. Puncture the key K_b at the index i; call this $K_b \setminus \{i\}$. Generate the obfuscated circuit as follows:

$$\widehat{\mathbf{H_{8.i}}.G_b} = \mathsf{iO.Obf}\left(\mathbf{H_8}.G_{b,\mathsf{OT}_2}[i, \mathsf{CRS_{OT}}, \mathsf{ch}_b, K_b \setminus \{i\}]; \mathsf{r}_b\right) \ ,$$

where $\mathbf{H_8}.G_{b,\mathsf{OT}_2}$ is defined in Circuit 6.13.

Circuit 6.13: $\mathbf{H_8}.G_{b,\mathsf{OT}_2}[i, \mathsf{CRS_{OT}}, \mathsf{ch}_b, K \setminus \{i\}](\mathsf{msg}_R)$:

- **Input:** msg_R.
- **Hardwired value:** index i, $\mathsf{CRS_{OT}}$, ch_b, a punctured pseudorandom key $K \setminus \{i\}$.
- **The circuit:**
  1. If $\mathsf{msg}_R > i$:
     - Let $\left(\mathbf{m}_0, \mathbf{m}_1, \{\mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}\}_{\tau_0,\tau_1 \in \{0,1\}}\right) \leftarrow \mathsf{PRF}_K(\mathsf{msg}_R)$.

- For every $\tau_1, \tau_2 \in \{0,1\}$, compute $\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \mathbf{m}_1 \oplus \tau_1^n, \mathsf{msg}_R; \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1})$.
- **Output:** $(\mathbf{m}_0 \oplus \mathbf{m}_1 \oplus \mathsf{ch}_b, \{\mathsf{msg}_S\}_{\tau_0,\tau_1})$.

2. If $\mathsf{msg}_R \leq i$:
   - Let $\left(\mathbf{m}_0, \mathbf{m}_1, \{\mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}\}_{\tau_0,\tau_1 \in \{0,1\}}\right) \leftarrow \mathsf{PRF}_K(\mathsf{msg}_R)$.
   - For every $\tau_1, \tau_2 \in \{0,1\}$, compute $\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \mathbf{m}_1 \oplus \tau_1^n, \mathsf{msg}_R; \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1})$.
   - **Output:** $(\mathbf{m}_0 \oplus \mathbf{m}_1, \{\mathsf{msg}_S\}_{\tau_0,\tau_1})$.

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{8.i}}$.

$\mathbf{H_9}$: We instead further modify the generation of the common reference string CRS as follows. Generate the obfuscated circuit as follows:

$$\widehat{\mathbf{H_9}.G_b} = \mathsf{iO.Obf}\left(\mathbf{H_9}.G_{b,\mathsf{OT}_2}[\mathsf{CRS}_{\mathsf{OT}}, K_b; \mathsf{r}_b]\right) \ ,$$

where $\mathbf{H_9}.G_{b,\mathsf{OT}_2}$ is defined in Circuit 6.14.

Circuit 6.14: $\mathbf{H_8}.G_{b,\mathsf{OT}_2}[\mathsf{CRS}_{\mathsf{OT}}, K](\mathsf{msg}_R)$:

- **Input:** msg_R.
- **Hardwired value:** index i, $\mathsf{CRS}_{\mathsf{OT}}$, a PRF key K.
- **The circuit:**
  1. Let $\left(\mathbf{m}_0, \mathbf{m}_1, \{\mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1}\}_{\tau_0,\tau_1 \in \{0,1\}}\right) \leftarrow \mathsf{PRF}_K(\mathsf{msg}_R)$.
  2. For every $\tau_1, \tau_2 \in \{0,1\}$, compute $\mathsf{msg}_S^{\tau_0,\tau_1} = \mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{m}_0 \oplus \tau_0^n, \mathbf{m}_1 \oplus \tau_1^n, \mathsf{msg}_R; \mathsf{r}_{\mathsf{OT}_2}^{\tau_0,\tau_1})$.
  3. **Output:** $(\mathbf{m}_0 \oplus \mathbf{m}_1, \{\mathsf{msg}_S\}_{\tau_0,\tau_1})$.

Let the probability that $\mathcal{A}$ on input the modified CRS outputs $(b, \mathsf{receiver}, \mathsf{ch})$ s.t. $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})$ is $\delta_{\mathbf{8.i}}$.

Indistinguishability of Hybrids. Let iO be $\varepsilon_{\mathsf{iO}}$-secure. Let PRF be $\varepsilon_{\mathsf{PRF}}$-secure. Let OT be $\varepsilon_{\mathsf{OT}}$-secure. Finally, let NIWI be $\varepsilon_{\mathsf{NIWI}}$-secure. We prove the indistinguishability of hybrids as follows.

- From the witness-indistinguishability property of $\pi^{(R)}$, and from the indistinguishability of modes in the CRS of the underlying OT protocol, it follows that $|\delta - \delta_{\mathbf{1}}| \leq \varepsilon_{\mathsf{NIWI}} + \varepsilon_{\mathsf{OT}}$.
- Note that the circuit G_{b,OT_2} is functionally equivalent to $\mathbf{H_{2.0}}.G_{b,\mathsf{OT}_2}$. Thus, from the security of iO, it follows that $|\delta_{\mathbf{1}} - \delta_{\mathbf{2.0}}| \leq \varepsilon_{\mathsf{iO}}$.
- For every $i \in \{0, \ldots, 2^L - 1\}$, where L is the length of the message msg_R it holds that:
  - From the security of PRF, it follows that $|\delta_{\mathbf{2.i}} - \delta_{\mathbf{3.i}}| \leq \varepsilon_{\mathsf{PRF}}$.
  - From the simulation security of OT against a malicious receiver, it follows that $|\delta_{\mathbf{3.i}} - \delta_{\mathbf{4.i}}| \leq \varepsilon_{\mathsf{OT}}$.
  - From the perfect security of one-time pad, it holds that $\delta_{\mathbf{4.i}} = \delta_{\mathbf{5.i}}$.
  - From the sender security of OT, it follows that $|\delta_{\mathbf{5.i}} - \delta_{\mathbf{6.i}}| \leq \varepsilon_{\mathsf{OT}}$.
  - From the security of PRF, it follows that $|\delta_{\mathbf{6.i}} - \delta_{\mathbf{7.i}}| \leq \varepsilon_{\mathsf{PRF}}$.
  - Note that the circuit $\mathbf{H_8}.G_{b,\mathsf{OT}_2}$ is functionally equivalent to $\mathbf{H_2}.G_{b,\mathsf{OT}_2}$. From the security of iO, it follows that $|\delta_{\mathbf{7.i}} - \delta_{\mathbf{8.i}}| \leq \varepsilon_{\mathsf{PRF}}$.

– The circuit $\mathbf{H.8}.G_{b,\mathsf{OT}_2}$ for i is functionally equivalent to $\mathbf{H_2}.G_{b,\mathsf{OT}_2}$ for $i+1$. From the security of iO, it follows that $|\delta_{\mathbf{8.i}} - \delta_{\mathbf{2.i+1}}| \leq \varepsilon_{\mathsf{iO}}$.

- The circuit $\mathbf{H.8}.G_{b,\mathsf{OT}_2}$ for $i = 2^L$ is functionally-equivalent to the circuit $\mathbf{H.9}.G_{b,\mathsf{OT}_2}$. From the security of iO, it follows that $|\delta_{\mathbf{8.2^L}} - \delta_{\mathbf{9}}| \leq \varepsilon_{\mathsf{iO}}$.

Thus, we have $O(2^L(\varepsilon_{\mathsf{NIWI}} + \varepsilon_{\mathsf{iO}} + \varepsilon_{\mathsf{OT}} + \varepsilon_{\mathsf{PRF}})) \geq |\delta - \delta_{\mathbf{9}}|$, where L is the length of msg_R.

From the security of one-way function to argue that $\delta_{\mathbf{9}}$ is negligible, where L is the length of msg_R. Moreover, setting $\varepsilon_{\mathsf{iO}}, \varepsilon_{\mathsf{OT}}, \varepsilon_{\mathsf{PRF}}$ to be $\frac{1}{2^{L+\lambda}}$ we have that δ is negligible.

□

□

7 Malicious Authority Security for Secure Two-Party Computation

In this section, we investigate malicious authority security for secure two party computation. We first show that secure two party computation for all functions is impossible. We then restrict our attention to specific class of functions in which malicious authority security can be achieved.

7.1 Impossibility Result: Malicious Authority Security for General Two-Party Computation is Impossible

Lemma 7.1. *There exists a two-party functionality F such that for any secure computation protocol for F between parties P_1 and P_2, the following events cannot simultaneously hold:*

- *P_i, for some $i \in \{1, 2\}$ receives the output of the protocol and,*
- *The following properties are satisfied: (i) defamation-free property and, (ii) accountability holds when the malicious authority corrupts the party P_i.*

Proof. Let F be an identity two-party functionality. Suppose there exists a secure two-party computation protocol Π with malicious authority security for F.

Suppose only P_1 receives the output; the other case is symmetrical. We now claim that one of the two properties – accountability when the malicious authority corrupts P_1 or defamation-free – is violated.

Lets suppose that accountability property is satisfied. This means that there exists an extractor $\mathcal{E}$ associated with the accountability property. We design an algorithm $\mathcal{B}$ as follows: for $i = 1, \ldots, q$,

- It sends $\left(x_1^{(i)}, r_1^{(i)}\right)$ to $\mathcal{E}$, where $x_1^{(i)}, r_1^{(i)}$ is sampled uniformly at random.
- In return, $\mathcal{E}$ sends a transcript trans_i.

Recover the output $y^{(1)} = (x_1^{(1)}, (x_2^*)^{(1)})$ from the transcript trans_1. Note that here we crucially use the fact that the view of the first party contains the output of the function. Send $(x_2^*)^{(1)}$ to $\mathcal{E}$. $\mathcal{E}$ outputs the evidence τ. Finally, $\mathcal{B}$ outputs τ.

We claim that $\mathcal{B}$ violates defamation-free property. Since $\mathcal{E}$ produces a τ such that Judge on input CRS and τ outputs $\mathsf{corrupted}$ with non-negligible probability ε, we have that $\mathcal{B}$ violates defamation-free property with the same non-negligible probability ε. This proves that the defamation-free and accountability properties against P_1 cannot simultaneously hold. □

7.2 The Basic Protocol

In this section, we construct a two-message maliciously secure two-party computation protocol Π to satisfy the accountability and defamation-free properties. We only consider the setting when only one of the parties receives the output. We call the party who receives the output as the receiver and the other party to be the sender. For simplicity we assume that the sender's input has the same length as the receiver's input, and the length is denoted as $\ell = \ell(\lambda)$. Generalizing to arbitrarily (sufficiently long) length is straightforward.

As shown, not all functions can be computed with malicious authority security. In the following, we restrict our attention for the following class of functions.

The class of functions. For $\ell > \lambda$, we let $\mathcal{F}$ be a family of all functions over $F : \{0,1\}^\ell \times \mathcal{Y} \to \{0,1\}^\ell$ such that

$$F\left((x_1, \ldots, x_\ell), (y_1, \ldots, y_\ell)\right) = g\left(\{x_i\}_{y_i=1}\right) \ ,$$

for some function g. Namely, whenever $y_i = 0$, then the x_i does not affect the output. The set $\mathcal{Y} \subset \{0,1\}^\ell$ contains all elements with hamming weight at most $\ell - \lambda$.

We first show how generic protocol for computing the function F in Section 7.2. In Section 7.3 we show how to enhance the security of this protocol and achieve malicious authority security.

Tools. We use the following tools in the construction:

- A rerandomizable maliciously secure $n \times \mathsf{OT}$ protocol in the CRS model. Denote the protocol as $\mathsf{OT}^n = (\mathsf{GenCRS}, \mathsf{OT}_1^n, \mathsf{OT}_2^n, \mathsf{Output})$. In case of a bit OT ($n = 1$), to avoid confusion with the receiver's first message OT_1, we write OT^{bit}.
- A rerandomizable garbled circuit $\mathsf{Garble} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Sim}, \mathsf{Rerand})$, as in Section 3.5. Henceforth, let λ be the length of the labels.
- A pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\mathsf{poly}(\lambda)}$.
- A non-interactive witness indistinguishability proof system (Definition 3.2) denoted by $\Pi_{\mathsf{NIWI}}^{(S)} = (\mathsf{Prove}, \mathsf{Verify})$ for the following language:
  1. **Statement:** $(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S, \mathsf{s}_{\mathsf{out}})$,
  2. **Witness:** $\left(\mathsf{r}_{\mathsf{Garble}}, \mathbf{x}, \mathsf{r}^x_{\mathsf{OT}_2}, \mathsf{r}^y_{\mathsf{OT}_2}, \mathsf{s}_{\mathsf{in}}\right)$,
  3. Such that one of the following conditions hold:
     (a) All of the following hold:
        i. $\left(\mathsf{GC}, \ \left\{K^{(x)}_{i,0}, K^{(x)}_{i,1}, K^{(y)}_{i,0}, K^{(y)}_{i,1}\right\}_{i[\ell]}\right) \leftarrow \mathsf{Garble}\left(1^\lambda, C; \mathsf{r}_{\mathsf{Garble}}\right)$, and
        ii. $\mathsf{msg}_S = \left\{\mathsf{OT}_2^{2\lambda+1}\left(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R^{(x)}[i], \left(K^{(x)}_{i,x_i} \| sh^0_i \| K^{(y)}_{i,0}, \ K^{(x)}_{i,x_i} \| sh^1_i \| K^{(y)}_{i,1}; \mathsf{r}^x_{\mathsf{OT}_2}[i]\right)\right)\right\}_{i \in [\ell]}$
     (b) Or, $\mathsf{s}_{\mathsf{out}} = \mathsf{PRG}(\mathsf{s}_{\mathsf{in}})$.

Looking ahead, when computing the function with malicious authority security, we will add one more possible witness for the language $\Pi_{\mathsf{NIWI}}^{(S)}$. This will correspond to obtaining the message msg_S as a rerandomization of the CRS.

Protocol 7.2: Secure Two-Party Computation in the CRS model

$\mathsf{GenCRS}(1^\lambda)$:

1. Run $\mathsf{OT}_{\mathsf{CRS}} \leftarrow \mathsf{OT}.\mathsf{GenCRS}(1^\lambda)$ to obtain $\mathsf{CRS}_{\mathsf{OT}}$.
2. Compute $\mathsf{s}_{\mathsf{out}} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.

3. Output $\mathsf{CRS} = (\mathsf{CRS}_{\mathsf{OT}}, \mathsf{s}_{\mathsf{out}})$.

The protocol:

- **The receiver:** On common reference string $\mathsf{CRS} = (\mathsf{CRS}_{\mathsf{OT}}, \mathsf{s}_{\mathsf{out}})$ and input $\mathbf{y} = (y_1, \ldots, y_\ell)$ compute the following:
  1. Compute $\mathsf{msg}_R = \mathsf{OT}_1^{\ell\cdot(2\lambda+1)}(\mathsf{CRS}_{\mathsf{OT}}, \mathbf{y}) = (\mathsf{OT}_1^{2\lambda+1}(\mathsf{CRS}_{\mathsf{OT}}, y_1), \ldots, \mathsf{OT}_1^{2\lambda+1}(\mathsf{CRS}_{\mathsf{OT}}, y_\ell))$.

  Send msg_R to the sender.
- **The sender:** On common reference string $\mathsf{CRS} = (\mathsf{CRS}_{\mathsf{OT}}, \mathsf{s}_{\mathsf{out}})$, input $\mathbf{x} = (x_1, \ldots, x_\ell)$, circuit C, and message msg_R received from the receiver, compute the following:
  1. Compute $\left(\mathsf{GC}, \left\{K_{i,0}^{(x)}, K_{i,1}^{(x)}, K_{i,0}^{(y)}, K_{i,1}^{(y)}\right\}_{i\in[\ell]}\right) \leftarrow \mathsf{Garble.Gen}\left(1^\lambda, C; \mathsf{r}_{\mathsf{Garble}}\right)$.
  2. For every $i \in [\ell]$, sample sh_i^0, sh_i^1 uniformly at random conditioned on $sh_i^0 \oplus sh_i^1 = x_i$.
  3. Compute $\mathsf{msg}_S = \left\{\mathsf{OT}_2^{2\lambda+1}\left(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R[i], \left(K_{i,0}^{(x)} \| sh_i^0 \| K_{i,0}^{(y)}, \quad K_{i,x_i}^{(x)} \| sh_i^1 \| K_{i,1}^{(y)}\right)\right)\right\}_{i\in[\ell]}$. Let $\mathsf{r}_{\mathsf{OT}_2}$ be the randomness used in computing these messages.
  4. Compute $\Pi_{\mathsf{NIWI}}^{(S)}.\mathsf{Prove}$ on instance $(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S, \mathsf{s}_{\mathsf{out}})$ with the following witness: $(\mathsf{r}_{\mathsf{Garble}}, \mathbf{x}, \mathsf{r}_{\mathsf{OT}_2}, \bot)$. Let the resulting proof be $\pi^{(S)}$.

  The sender sends $\left(\mathsf{GC}, \mathsf{msg}_S, \pi^{(S)}\right)$ to the receiver.
- **The receiver:**
  1. Verify that $\Pi_{\mathsf{NIWI}}^{(S)}.\mathsf{Verify}\left((\mathsf{CRS}_{\mathsf{OT}}, \mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S, \mathsf{s}_{\mathsf{out}}), \pi^{(S)}\right) = 1$. If it does not hold, then abort.
  2. From msg_S, recover the labels sent in OT_2 according to the choice bits $\mathbf{y}$ for every $i \in [\ell]$ and discard sh_{y_i}. Let $K_i^{(x)}, K_i^{(y)}$ denote the received labels for every $i \in [\ell]$.
  3. Output the result of $\mathsf{Garble.Eval}(\mathsf{GC}, (\{K_i^{(x)}, K_i^{(y)}\}_{i\in[\ell]}))$.

Lemma 7.3. *Assuming the security of maliciously secure oblivious transfer* OT*, pseudorandom generator* PRG*, non-interactive witness-indistinguishable proof, Protocol 7.2 is a maliciously secure two-party protocol in the CRS model for any function* $F \in \mathcal{F}$ *where only the receiver receives the output.*

We show simulation security separately against malicious receiver and malicious sender.

Simulation against malicious receiver. Suppose $\mathcal{A}$ be a PPT malicious receiver. We construct a PPT simulator Sim that simulates the view of $\mathcal{A}$. Before we describe Sim, we rely upon the simulators of the OT protocol and garbling scheme. We denote $\mathsf{Sim}_{\mathsf{OT}}$ to be the simulator of the OT protocol defined for malicious receivers and $\mathsf{Garble.Sim}$ be the simulator of the garbling scheme.

Description of Sim:

- CRS generation procedure:
  1. Compute the CRS generation algorithm of $\mathsf{Sim}_{\mathsf{OT}}$ to obtain $\mathsf{CRS}_{\mathsf{OT}}^*$ along with a trapdoor $\mathsf{td}_{\mathsf{OT}}$, and sample $\mathsf{s}_{\mathsf{in}} \leftarrow \{0,1\}^\lambda$ and set $\mathsf{s}_{\mathsf{out}} = \mathsf{PRG}(\mathsf{s}_{\mathsf{in}})$.

  Output the following: $\mathsf{CRS} = (\mathsf{CRS}_{\mathsf{OT}}^*, \mathsf{s}_{\mathsf{out}})$.

- Sim receives msg_R from $\mathcal{A}$. It also runs $\mathsf{Sim}_{\mathsf{OT}}$ on every $\mathsf{OT}_1^{2\lambda+1}(\cdot)$ message in msg_R to extract the receiver's input $\mathbf{y}^* = (y_1^*, \ldots, y_\ell^*)$.
  It then submits $\mathbf{y}^*$ to the trusted functionality and it receives the output $\mathbf{z} = F(\mathbf{x}, \mathbf{y}^*)$, where $\mathbf{x}$ is the input that the honest sender has sent to the trusted party.
- Sim then computes $\mathsf{Garble.Sim}(1^\lambda, C, \mathbf{z})$ to obtain the simulated garbled circuit $\widetilde{\mathsf{GC}}$ along with simulated wire labels $\{\widetilde{K}_i^{(x)}\}_{i\in[\ell]}, \{\widetilde{K}_i^{(y)}\}_{i\in[\ell]}$.
  It then computes the following messages:
  - Compute $\mathsf{msg}_S = \left\{\mathsf{OT}_2^{2\lambda+1}\left(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}_R[i], \left(\widetilde{K}_i^{(x)} \| sh_i \| \widetilde{K}_i^{(y)},\ \widetilde{K}_i^{(x)} \| sh_i \| \widetilde{K}_i^{(y)}\right)\right)\right\}_{i\in[\ell]}$, where $sh_i \leftarrow \{0,1\}$.

  Compute $\Pi_{\mathsf{NIWI}}^{(S)}.\mathsf{Prove}$ on the instance $(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S, \mathsf{s}_{\mathsf{out}})$ with the witness $(\bot, \cdots, \bot, \mathsf{s}_{\mathsf{in}})$ to obtain the proof $\pi^{(S)}$.
  Sim sends $\left(\mathsf{GC}, \mathsf{msg}_S, \pi^{(S)}\right)$ to $\mathcal{A}$.

Claim 7.4. *The output of the ideal world is computationally indistinguishable from the output of the real world.*

Proof. We prove this by a hybrid argument.

1. $\mathbf{H}_1$: The output of this hybrid is the output of the real world execution – the output of the honest sender (which is empty) and the output of the adversary.
2. $\mathbf{H}_2$: Here we change GenCRS to set $\mathsf{s}_{\mathsf{out}}$ to be an output of PRG for a random seed s_{in} instead of a uniform value in $\{0,1\}^{\mathsf{poly}(\lambda)}$.
3. $\mathbf{H}_3$: In this hybrid, the proof $\pi^{(S)}$ is generated using the witness s_{in}. That is, compute $\pi^{(S)}$ using $\Pi_{\mathsf{NIWI}}^{(S)}.\mathsf{Prove}$ on input instance $(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S, \mathsf{s}_{\mathsf{out}})$ with witness $(\bot, \cdots, \bot, \mathsf{s}_{\mathsf{in}})$; the notation is the same as in the description of the protocol. The rest of the hybrid is the same as before.
4. $\mathbf{H}_4$: In this hybrid, OT protocol in the execution of Π. The experiment receives msg_R from $\mathcal{A}$ and runs $\mathsf{Sim}_{\mathsf{OT}}$. Also, the receiver's input $\mathbf{y}^*$ is extracted by $\mathsf{Sim}_{\mathsf{OT}}$, using $\mathsf{td}_{\mathsf{OT}}$, from the message of the receiver. The message msg_S is computed as in the protocol.
5. $\mathbf{H}_5$: In this hybrid, we no longer know $\mathbf{x}$ for interacting with the adversary, and we have an interaction with a trusted party. We extract $\mathbf{y}^*$ as in the previous hybrid and send it to the trusted party to obtain $\mathbf{z} = F(\mathbf{x}, \mathbf{y}^*)$ where now $\mathbf{x}$ is the input that the honest sender has sent to the trusted party. We use $\mathsf{Garble.Sim}$ to obtain the simulated garbled circuit $\widetilde{\mathsf{GC}}$ along with simulated wire labels $\{\widetilde{K}_i^{(x)}\}_{i\in[\ell]}, \{\widetilde{K}_i^{(y)}\}_{i\in[\ell]}$ and we send to the adversary the message msg_S as in the simulated execution.
6. $\mathbf{H_6}$: The output of this hybrid is the output of the ideal world.

The computational indistinguishability of $\mathbf{H}_1$ and $\mathbf{H}_2$ follows from the security of the pseudorandom generator. The computational indistinguishability of $\mathbf{H}_2$ and $\mathbf{H}_3$ follow form the witness-indistinguishability property of $\Pi_{\mathsf{NIWI}}^{(S)}$. The indistinguishability of $\mathbf{H}_3$ and $\mathbf{H}_4$ follows by a hybrid argument on the simulation security of OT against malicious receivers. The indistinguishability of $\mathbf{H_4}$ and $\mathbf{H_5}$ follows from the simulation security of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$. $\mathbf{H_5}$ and $\mathbf{H_6}$ are identical. □

Simulation Security against malicious senders. Suppose $\mathcal{A}$ is a PPT malicious sender. We construct a PPT simulator Sim that simulates the view of $\mathcal{A}$. Before we describe Sim, we rely upon the simulators of the OT protocol and garbling scheme. We denote $\mathsf{Sim}_{\mathsf{OT}}$ to be the simulator of

the OT protocol defined for malicious senders.

Description of Sim:

1. CRS generation procedure:
   (a) Compute the CRS generation algorithm of $\mathsf{Sim}_{\mathsf{OT}}$ to obtain $\mathsf{CRS}^*_{\mathsf{OT}}$ along with a trapdoor $\mathsf{td}_{\mathsf{OT}}$.
   (b) Sample $\mathsf{s}_{\mathsf{out}}$ uniformly at random from $\{0,1\}^{\mathsf{poly}(\lambda)}$.

   Output the following: $\mathsf{CRS} = (\mathsf{CRS}^*_{\mathsf{OT}}, \mathsf{s}_{\mathsf{out}})$.
2. Sim executes $\mathsf{Sim}_{\mathsf{OT}}$ to compute msg_R.
3. Upon receiving the message $\left(\mathsf{GC}, \mathsf{msg}_S, \pi^{(S)}\right)$, do the following:
   - Compute $\Pi^{(S)}_{\mathsf{NIWI}}.\mathsf{Verify}$ on input the instance $(\mathsf{CRS}, \mathsf{GC}, \mathsf{msg}_S, \mathsf{msg}_R, \mathsf{s}_{\mathsf{out}})$ and the proof $\pi^{(S)}$. If $\Pi^{(S)}_{\mathsf{NIWI}}.\mathsf{Verify}$ outputs reject, then abort the execution. Otherwise, continue.
   - Compute $\mathsf{Sim}_{\mathsf{OT}}$ on input msg_S and trapdoor $\mathsf{td}_{\mathsf{OT}}$ to extract the input

   $$\left\{(K^{(x)}_{i,0} \| sh^0_i \| K^{(y)}_{i,0}, \quad K^{(x)}_{i,x_i} \| sh^1_i \| K^{(y)}_{i,1})\right\} .$$

   Extract $x_i = sh^0_i \oplus sh^1_i$. If $x_i = 0$ then if $K^{(x)}_{i,x_i} \neq K^{(x)}_{i,0}$ then abort. If $x_i = 1$ then if $K^{(x)}_{i,x_i} = K^{(x)}_{i,0}$ then abort.
   Let $\mathbf{x} = (x_1, \ldots, x_\ell)$.

   Finally, Sim sends $\mathbf{x}$ to the trusted party.

Claim 7.5. *The output of the ideal world is computationally indistinguishable from the output of the real world.*

Proof. We prove this by a hybrid argument.

1. $\mathbf{H}_1$: The output of this hybrid is the output of the real world, consisting of the view of the corrupted sender and the output of the honest receiver.
2. $\mathbf{H}_2$: In this hybrid, we simulate the OT protocol in the execution of Π. The rest of the hybrid is the same as before. That is, $\mathsf{CRS}_{\mathsf{OT}}$, along with trapdoor $\mathsf{td}_{\mathsf{OT}}$, is generated using $\mathsf{Sim}_{\mathsf{OT}}$. All the $\mathsf{OT}_1(\cdot)$ messages are simulated by $\mathsf{Sim}_{\mathsf{OT}}$. That is, execute $\mathsf{Sim}_{\mathsf{OT}}$ to compute msg_R. Finally, upon receiving the second round message from the corrupted sender, compute $\mathsf{Sim}_{\mathsf{OT}}$ to extract the input $(K^{(x)}_{i,0} \| sh^0_i \| K^{(y)}_{i,0}, \quad K^{(x)}_{i,x_i} \| sh^1_i \| K^{(y)}_{i,1})$ using the trapdoor $\mathsf{td}_{\mathsf{OT}}$. Let $x_i = sh^0_i \oplus sh^1_i$. If $x_i = 0$ then if $K^{(x)}_{i,x_i} \neq K^{(x)}_{i,0}$ then abort. If $x_i = 1$ then if $K^{(x)}_{i,x_i} = K^{(x)}_{i,0}$ then abort. Let $\mathbf{x} = (x_1, \ldots, x_\ell)$ and send it to the trusted party. The output of the honest receiver is then $C(\mathbf{x}, \mathbf{y})$ where $\mathbf{y}$ is its input (i.e., the receiver does not compute the garbled circuit but instead evaluate the circuit directly).
3. $\mathbf{H}_3$: The output of this hybrid is the output of the ideal world.

To argue the computational indistinguishability of $\mathbf{H_1}$ and $\mathbf{H_2}$, we need to argue the following: (a) the output distribution of the hybrid receiver in $\mathbf{H_1}$ is identical to the output distribution of the hybrid receiver in $\mathbf{H_2}$ and, (b) the view of $\mathcal{A}$ in $\mathbf{H_1}$ is computationally indistinguishable from the view of $\mathcal{A}$ in $\mathbf{H_2}$. We note that (a) follows from the perfect soundness of $\Pi^{(R)}_{\mathsf{NIWI}}$, and the fact that for every $y_i = 0$ the input x_i does not affect the output. Furthermore, (b) follows from a hybrid argument on the simulation security of OT against malicious senders.

The only difference between $\mathbf{H}_3$ and $\mathbf{H}_2$ is in the computation of the output of the receiver. In $\mathbf{H_2}$, the hybrid receiver extracts $\mathbf{x}$ and computes $C(\mathbf{x}, \mathbf{y})$ with its input $\mathbf{y}$. In $\mathbf{H_3}$, the simulator extractors $\mathbf{x}$ and sends it to the trusted functionality and the receiver receives $F(\mathbf{x}, \mathbf{y})$ from the trusted functionality. Since C computes F, the output distributions of $\mathbf{H_2}$ and $\mathbf{H_3}$ are identical. □

7.3 Extending to Malicious Authority Security

To achieve malicious authority security, we change the protocol and incorporate more information in the CRS. Specifically, since the only message that the receiver sends in the protocol is OT_1, the case of a corrupted sender is easy and is identical to Construction 6.4. The case of a corrupted receiver is more involved, and we add some OT transcripts that embeds some challenge, and allows the extractor to create the message that the sender sends to the receiver in the protocol. Moreover, we allow the sender to submit also a message that was generated by the extractor as the extractor generates its message as a re-randomization of elements in the CRS and not according to the underlying protocol. The way the extractor generated the transcript is described in Procedure 7.9, and we just refer in the description of the language to that the generated message is an output of that extractor. Procedure 7.9 is specified in the proof of Theorem 7.7.

Corrupted receievr. For handling the corrupted receiver, let ch be be the challenge embedded in the CRS (of λ bits). The extractor has to generate a transcript that embeds ch in λ positions of the OT_2 messages. We embed this information in the keys $K^{(x)}_{i,x_i}$ that the sender sends in its OT message (recall Step 3 in the base protocol 7.2):

$$K^{(x)}_{i,0} \parallel sh^0_i \parallel K^{(y)}_{i,0}, \quad K^{(x)}_{i,x_i} \parallel sh^1_i \parallel K^{(y)}_{i,1} \tag{5}$$

Recall that according to the class of functions we support, sending $K^{(x)}_{i,x_i}$ instead of $K^{(x)}_{i,1}$ does no affect correctness, as whenever $y_i = 0$ the value of the bit x_i does not affect the output. Thus, whenever $y_i = 1$, the label $K^{(x)}_{i,x_i}$ is important for the computation and is revealed to the receiver, and we cannot embed ch in it. For this OTs correspond to $y_i = 1$, the extractor just generates the OT_2 message as in a regular execution.

However, for exactly λ positions for which $y_i = 0$, the extractor has to learn the jth bit (for $j \in [\lambda]$) of the challenge ch. This is done by replacing the $K^{(x)}_{i,x_i}$ label with $K^{(x)}_{i,\mathsf{ch}[j]+\Delta_i}$ for some shift Δ_i of its choice in the OT message. I.e., the extractor generates a transcript for input $x_i = \mathsf{ch}[j]+\Delta_i$ that it does not know. Recall that the label $K^{(x)}_{i,x_i}$ is not revealed in an honest execution (as $y_i = 0$), however, if the authority ever opens the transcript it would have to open it with respect to input where $x_i = \mathsf{ch}[j] + \Delta_i$, and so the extractor would be able to recover the jth bit of the challenge.

To generate such a transcript, we add to the CRS:

$$\beta_{j,m,\ell_0,\ell_1} := \mathsf{OT}^{bit}(0, (\ell_0, 1 - (\mathsf{ch}[j] \oplus m) \cdot \ell_0 + (\mathsf{ch}[j] \oplus m) \cdot \ell_1)) \ .$$

(As previously, we will actually have to generate two challenges $\mathsf{ch}_0, \mathsf{ch}_1$ and use NIWI to prove that one of the branches was honestly generated. We omit the branch $b \in \{0, 1\}$ in the above description for brevity). This enables the extractor to generate, for every two label keys of its choice, K_0, K_1, and for a shift $\Delta_i \in \{0, 1\}$, a message that corresponds to OT where the input of the receiver is 0 and the input of the sender is:

$$\begin{cases} K_0 & \text{if } \mathsf{ch}[j] \oplus \Delta_i = 0 \\ K_1 & \text{if } \mathsf{ch}[j] \oplus \Delta_i = 1 \end{cases} ,$$

or in other words, the label is $K_{\mathsf{ch}[j]\oplus\Delta_i}$. We therefore can generate the first λ part of the message (bit-by-bit), i.e., the first λ bits of the OT message (recall Eq. 5):

$$\underline{K^{(x)}_{i,0}} \parallel sh^0_i \parallel K^{(y)}_{i,0}, \quad \underline{K^{(x)}_{i,x_i}} \parallel sh^1_i \parallel K^{(y)}_{i,1}$$

Next, we need to generate the OT message corresponds to sh^0_i, sh^1_i, and recall that $sh^0_i \oplus sh^1_i = x_i$, however, the extractor does not known x_i $(= \mathsf{ch}[j] + \Delta_i)$. We therefore also add to the CRS the following information:

$$\gamma_{j,m_0,m_1} := \mathsf{OT}(0, (m_0, \mathsf{ch}[j] \oplus m_1)) \ .$$

Thus, by choosing a random $sh^0_i \leftarrow \{0,1\}$ and taking $\gamma_{j,sh^0_i,\Delta_i}$, the extractor essentially obtains OT^{bit} message that corresponds to message:

$$\mathsf{OT}^{bit}(0, (sh^0_i, sh^0_i \oplus (\Delta_i \oplus \mathsf{ch}[j]))) = \mathsf{OT}^{bit}(0, (sh^0_i, sh^1_i)) \ ,$$

for $sh^0_i \oplus sh^1_i = \Delta_i \oplus \mathsf{ch}[j]$. Finally, for generating the last part of the message (recall Eq. 5):

$$K^{(x)}_{i,0} \parallel sh^0_i \parallel \underline{K^{(y)}_{i,0}}, \quad K^{(x)}_{i,x_i} \parallel sh^1_i \parallel \underline{K^{(y)}_{i,1}}$$

the extractor does not need any information from the CRS, since this is independent of the challenge ch. Procedure 7.9 shows how the extractor generates its message as a function of the CRS, the message it receives from the receiver, and inputs $\mathbf{y}$ (for the receiver) and $\mathbf{\Delta}$ (a "shift" for the sender).

Modifying the $\Pi^{(S)}_{\mathsf{NIWI}}$. The extractor can then obtain all this information from the CRS and re-randomize all the OT messages such that they look fresh and honestly generated. However, recall that in the protocol, the receiver checks that the OT was honestly generated, and the extractor cannot prove the NIWI statement without knowing the input its OT message corresponds to. Therefore, we add another branch to the NIWI language:

We define a non-interactive witness indistinguishability proof system (Definition 3.2) denoted by $\Pi^{(S)}_{\mathsf{NIWI}} = (\mathsf{Prove}, \mathsf{Verify})$ for the following language:

1. **Statement:** $(\mathsf{CRS}, \mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S, \mathsf{s}_{\mathsf{out}})$,
2. **Witness:** $\left(\mathsf{r}_{\mathsf{Garble}}, \mathbf{x}, \mathsf{r}^x_{\mathsf{OT}_2}, \mathsf{r}^y_{\mathsf{OT}_2}, \mathsf{s}_{\mathsf{in}}, b, \mathbf{y}, \mathbf{r}, \Delta, R\right)$,
3. Such that one of the following conditions hold:
   (a) All of the following hold:
      i. $\left(\mathsf{GC}, \left\{K^{(x)}_{i,0}, K^{(x)}_{i,1}, K^{(y)}_{i,0}, K^{(y)}_{i,1}\right\}_{i[\ell]}\right) \leftarrow \mathsf{Garble}\left(1^\lambda, C; \mathsf{r}_{\mathsf{Garble}}\right)$, and
      ii. $\mathsf{msg}_S = \left\{\mathsf{OT}_2\left(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{msg}^{(x)}_R[i], \left(K^{(x)}_{i,x_i} \| sh^0_i \| K^{(y)}_{i,0},\ K^{(x)}_{i,x_i} \| sh^1_i \| K^{(y)}_{i,1}; \mathsf{r}^x_{\mathsf{OT}_2}[i]\right)\right)\right\}_{i\in[\ell]}$.
   (b) Or, $\mathsf{s}_{\mathsf{out}} = \mathsf{PRG}(\mathsf{s}_{\mathsf{in}})$.
   (c) Or $(\mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S)$ is an output of Procedure 7.9 on input $(\mathsf{CRS}, b, \mathbf{y}, \mathbf{r}, \Delta)$ with randomness R.

We are now ready for a formal description of the construction.

Construction 7.6: Secure Two-Party Computation with Malicious Authority Security

---

$\mathsf{GenCRS}(1^\lambda)$:

1. Run $\mathsf{GenCRS}(1^\lambda)$ of Protocol 7.2 to obtain $(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{s}_{\mathsf{out}})$.

2. **Information for corrupted sender:** Identical to Construction 6.4.
   Let $(\mathsf{CRS}_{\mathsf{OT}}, (\Sigma_0, \boldsymbol{\alpha}_0), (\Sigma_1, \boldsymbol{\alpha}_1), \pi^{(S-CRS)})$ be the result of that step. We let $|\boldsymbol{\sigma}_0| = |\boldsymbol{\sigma}_1| = \lambda$.
3. **Information for corrupted receiver:**
   (a) For both $b = 0$ and $b = 1$:
      i. Choose random $\mathsf{ch}_b \in \{0,1\}^\lambda$ and compute $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch}_\mathsf{b})$.
      ii. Choose randomness R_b for the receiver.
      iii. For every $j \in [\lambda]$, masking $m \in \{0,1\}$, labels $\ell_0, \ell_1 \in \{0,1\}$, set:

$$\beta_{b,j,m,\ell_0,\ell_1} := \mathsf{OT}^{bit}\left(0, (\ell_0,\ (1 - (\mathsf{ch}_b[j] \oplus m)) \cdot \ell_0 + (\mathsf{ch}_b[j] \oplus m) \cdot \ell_1)\right) \ ,$$

      with respect to $\mathsf{CRS}_{\mathsf{OT}}$, using randomness S_{b,j,m,ℓ_0,ℓ_1} for the sender and the same randomness R_b for the receiver. We let $\boldsymbol{\beta}_b := \{\beta_{b,j,m,\ell_0,\ell_1}\}_{j\in[\lambda],m,\ell_0,\ell_1\in\{0,1\}}$, $\mathbf{S}_b := \{S_{b,j,m,\ell_0,\ell_1}\}_{j\in[\lambda],m,\ell_0,\ell_1\in\{0,1\}}$.
      iv. For every $j \in [\lambda]$, masking $m_0, m_1 \in \{0,1\}$ set:

$$\gamma_{b,j,m_0,m_1} := \mathsf{OT}^{bit}\left(0, (m_0,\ \mathsf{ch}_b[j] \oplus m_1)\right) \ ,$$

      with respect to $\mathsf{CRS}_{\mathsf{OT}}$, using randomness S_{b,j,m_0,m_1} for the sender and the same randomness R_b for the receiver.
      Let $\boldsymbol{\gamma}_b := \{\gamma_{b,j,m_0,m_1}\}_{j\in[\lambda],m_0,m_1\in\{0,1\}}$, $\mathbf{S}'_b = \{S_{b,j,m_0,m_1}\}_{j\in[\lambda],m_0,m_1\in\{0,1\}}$.
   (b) Using $\Pi^{(R-CRS)}_{\mathsf{NIWI}}$ for showing that either the transcript for $b = 0$ was generated correctly, or $b = 1$. That is, we use $\Pi^{(R-CRS)}_{\mathsf{NIWI}}$ for the following language:
   - **Instance:** $((\mathsf{CHR}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0, \mathsf{R}_0), (\mathsf{CHR}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_1, \mathsf{R}_1))$.
   - **Witness:** $((\mathsf{ch}_0, \mathbf{S}_0, \mathbf{S}'_0), (\mathsf{ch}_1, \mathbf{S}_1, \mathbf{S}'_1))$.
   - **Such that:** Either for $b = 0$ or for $b = 1$ it holds that:
      i. $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch}_\mathsf{b})$.
      ii. For every $j \in [\lambda]$, $m \in \{0,1\}$ and $\ell_0, \ell_1 \in \{0,1\}$, it holds that $\beta_{b,j,m,\ell_0,\ell_1}$ was generated as described using randomness S_{b,j,m,ℓ_0,ℓ_1} for the sender and R_b for the receiver.
      iii. For every $j \in [\lambda]$, $m_0, m_1 \in \{0,1\}$ it holds that $\boldsymbol{\gamma}_{b,j,m_0,m_1}$ was generated as described using randomness $S_{b,j,m}$ for the sender and R_b for the receiver.

   Use $\Pi^{(R-CRS)}_{\mathsf{NIWI}}$ to prove the statement $(\mathsf{CHR}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0, \mathsf{R}_0), (\mathsf{CHR}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_1, \mathsf{R}_1)$ using the witness $((\mathsf{ch}_0, \mathbf{S}_0, \mathbf{S}'_0), (\perp, \perp, \perp))$, and let $\pi^{(R-CRS)}$ be the resulting proof.
4. **Output:**

$$\mathsf{CRS} = \Big(\mathsf{CRS}_{\mathsf{OT}} \quad , \quad ((\Sigma_0, \boldsymbol{\alpha}_0, \mathbf{S}_0, \Sigma_1, \boldsymbol{\alpha}_1, \mathbf{S}_1), \pi^{(S-CRS)}) \qquad ((\mathsf{CHR}_0, \boldsymbol{\beta}_0, \mathsf{R}_0, \mathsf{CHR}_1, \boldsymbol{\beta}_1, \mathsf{R}_1), \pi^{(\mathsf{R-CRS})})\Big) \ . \tag{6}$$

The protocol. The implementation of the protocol is the same as Protocol 7.2, where the NIWI proof system $\Pi^{(S)}_{\mathsf{NIWI}}$ has also a third branch as described above.

Judge(CRS, τ).

1. Parse: CRS as in Eq. (6). Run $\Pi^{(R-CRS)}_{\mathsf{NIWI}}$.Verify to verify $\pi^{(R-CRS)}$ on the CRS was generated correctly, and run $\Pi^{(S-CRS)}_{\mathsf{NIWI}}$.Verify to verify that $\pi^{(S-CRS)}$ was generated correctly. If the verification did not pass, then output corrupted.

2. Otherwise, there are two options:
   (a) If $\tau = (b, \mathsf{sender}, \boldsymbol{\sigma}'_b)$, then verify that $\Sigma_b = f(\boldsymbol{\sigma}'_b)$. If so, output $\mathsf{corrupted}$.
   (b) If $\tau = (b, \mathsf{receiver}, \mathsf{ch}'_b)$, then verify that $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch}'_\mathsf{b})$. If so, output $\mathsf{corrupted}$.
3. Otherwise, output honest.

Theorem 7.7. *For every function $F \in \mathcal{F}$, Construction 7.6 has malicious security for the Construction 6.4 has malicious authority security with respect to the uniform distribution over the inputs.*

Proof. According to the definition of malicious authority security (Definition 4.2) we have to show simulatability of the OT protocol, accountability and defamation free.

Simulation. The construction is identical to Protocol 7.9, where we only change the language used in $\Pi^{(R)}_{\mathsf{NIWI}}$. We claim, however, that the adversary cannot use the third branch. In particular, the corrupted sender receives msg_R from the honest receiver. To use the third branch of $\Pi^{(R)}_{\mathsf{NIWI}}$, it has to find $\mathbf{y}, \mathbf{r}$ for which $\mathsf{msg}_R = \mathsf{OT}_1(\mathsf{CRS}_\mathsf{OT}, \mathbf{y}; \mathbf{r})$, which can occur with only negligible probability, from the receiver's security of oblivious transfer.

Accountability: sender. Fix a malicious authority $\mathcal{A}$, which chooses the inputs of the sender. We show that if there exists a non-negligible function $\epsilon_1(\cdot)$ for which

$$\Pr[\mathsf{Acc.Real}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda) = 1] \geq \epsilon_1(\lambda) \ ,$$

then there exists a non-negligible function $\epsilon_2(\cdot)$ such that

$$\Pr[\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda) = 1] \geq \epsilon_2(\lambda) \ .$$

The extractor Ext works as follows:

1. Obtain CRS^* from $\mathcal{A}$.
2. Check that CRS^* is well formed. I.e., run Step 1 in the Judge algorithm. If the CRS^* is corrupted, then output $\tau = \mathsf{CRS}^*$ and halt.
3. Choose a random $b \leftarrow \{0,1\}$, and run the following for q iterations where in the kth execution:
   (a) Sample a masking shift $\Delta^{(k)} = (\Delta_1, \ldots, \Delta_\ell)$.
   (b) Receive from the malicious authority $\mathcal{A}$ the input $\mathbf{x} = (x_1, \ldots, x_\ell)$ and randomness $\mathbf{r}$.
   (c) Generate $\hat{\mathsf{msg}}_R$ as follows. For every $i \in [\ell]$:
      i. For every $i \leq \lambda$: Set $\mathsf{msg}_R[i] = \boldsymbol{\alpha}_{b,\Delta_i}[i]$ and set $\mathsf{msg}^{(y)}_R[i] \leftarrow \mathsf{OT.Rerand}_1(\hat{\mathsf{msg}}_R[i])$.
      ii. For every $i > \lambda$: Set $\mathsf{msg}_R[i] = \mathsf{OT}_1(\mathsf{CRS}_\mathsf{OT}, \Delta_i)$.
   (d) Run the honest sender with input $\mathbf{x}$ and randomness $\mathbf{r}$ on message msg_R as generated by the extractor. Let msg_S be the message of the sender. Combine $\mathsf{trans}_k = (\mathsf{msg}_R, \mathsf{msg}_S)$ and send it to the malicious authority.
4. Let $(k, \mathbf{y}^{(k)})$ be the output of the adversary $\mathcal{A}$, where the input $\mathbf{y}^{(k)}$ represents the input of the receiver in the kth execution. Extract $\boldsymbol{\sigma}'$ from the first λ coordinates of $\mathbf{y}^{(k)}$ using Δ. If $f(\boldsymbol{\sigma}') = \Sigma_b$ then output $\tau = (b, \mathsf{sender}, \boldsymbol{\sigma}')$. Otherwise, output fail.

Assuming that the adversary succeeds in $\mathsf{Acc.Real}$ with probability $\epsilon_1(\lambda)$, then the extractor succeeds with probability close to $\epsilon_1(\lambda)/2$. To show that, we claim that the view of the adversary in

$\mathsf{Acc.Real}$ is identically distributed to its view in $\mathsf{Acc.Ext}$, conditioned on the event that the extractor guess the correct branch, i.e., b^* for which the malicious authority used its witness in $\Pi_{\mathsf{NIWI}}^{(S-CRS)}$.

Specifically, the view that the extractor produces correspond to the case where the receiver uses input $(\Delta_1 \oplus \boldsymbol{\sigma}_b[1], \ldots, \Delta_\lambda \oplus \boldsymbol{\sigma}_b[\lambda], \Delta_{\lambda+1}, \ldots, \Delta_\ell)$. Since $(\Delta_1, \ldots, \Delta_\ell)$, this is distributed as a uniform random input for the receiver. In the real execution, the same occurs. Moreover, when the malicious authority extracts the input of the receiver, clearly the extractor produces the correct output. Therefore, the extractor succeeds with probability $\epsilon_1(\lambda)/2$, which is non-negligible assuming that $\epsilon_1(\lambda)$ is non-negligible.

Accountability: receiver. Fix a malicious authority $\mathcal{A}$ which chooses the inputs of the receiver. We show that if there exists a non-negligible function $\epsilon_1(\cdot)$ for which

$$\Pr[\mathsf{Acc.Real}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda) = 1] \geq \epsilon_1(\lambda) \ ,$$

then there exists a non-negligible function $\epsilon_2(\cdot)$ such that

$$\Pr[\mathsf{Acc.Ext}_{\Pi,\mathcal{A},\mathcal{E},q}(\lambda) = 1] \geq \epsilon_2(\lambda) \ .$$

The extractor Ext works as follows:

1. Receive CRS^* from $\mathcal{A}$.
2. Check that CRS^* is well formed. I.e., run Step 1 in the Judge algorithm. If the CRS^* is corrupted, then output $\tau = \mathsf{CRS}^*$ and halt.
3. Choose a random $b \leftarrow \{0,1\}$, and run the following for q iterations where in the kth iteration:
   (a) Sample a masking shift $\boldsymbol{\Delta} = \Delta^{(k)} = (\Delta_1, \ldots, \Delta_\ell)$ uniformly at random for the input of the sender.
   (b) Receive from the malicious authority $\mathcal{A}$ the input $\mathbf{y}^{(k)} = \mathbf{y} = (y_1, \ldots, y_\ell)$ and randomness $\mathbf{r} = (\mathbf{r}[1], \ldots, \mathbf{r}[\ell])$ for the receiver.
   (c) Run Procedure 7.9 on input $(\mathsf{CRS}^*, b, \mathbf{y}, \mathbf{r}, \boldsymbol{\Delta})$ with some randomness R. Let $(\mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S)$ be the output.
   (d) Compute $\Pi_{\mathsf{NIWI}}^{(S)}.\mathsf{Prove}$ on instance $(\mathsf{CRS}, \mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S, \mathsf{s}_{\mathsf{out}})$ with witness: $(\bot, \ldots, b, \mathbf{y}, \mathbf{r}, \Delta, R)$. Let the resulting proof be $\pi^{(S)}$.
   (e) Send $\mathcal{A}$ the resulted transcript $\mathsf{trans}_k = (\mathsf{msg}_R, \mathsf{GC}, \mathsf{msg}_S, \pi^{(S)})$.
4. If $\mathcal{A}$ replies with some input $(k, \mathbf{x})$, then let $\mathbf{y} = \mathbf{y}^{(k)} = (y_1, \ldots, y_\ell)$. Extract ch' from the first λ positions in which $y_i = 0$ and using $\boldsymbol{\Delta}$. If $f(\mathsf{ch}') = \mathsf{CHR_b}$ then output $\tau = (b, \mathsf{receiver}, \mathsf{ch}')$.

Similarly to the previous case, we claim that the view of the adversary is indistinguishable assuming that the extractor chose the correct branch. This follows from the correctness of the rerandomization process, and the witness-indistinguishablity of $\Pi_{\mathsf{NIWI}}^{(S)}$. Therefore, if the adversary succeeds in $\mathsf{Acc.Real}$ with probability $\epsilon_1(\lambda)$, then it will succeeds in $\mathsf{Acc.Ext}$ with probability $\epsilon_1(\lambda)/2 - \mu(\lambda)$ for some negligible function $\mu(\lambda)$, which comes form the witness-indistinguishability security loss.

Defamation free. We show that for every non-uniform adversary $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for all λ:

$$\Pr\left[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}\right] \leq \mu(\lambda) \ ,$$

where $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$.

Fix an adversary $\mathcal{A}$. Since CRS is honestly generated, the check in Step 1 always passes. We thus can write:

$$\begin{aligned}\Pr\left[\mathsf{Judge}(\mathsf{CRS}, \mathcal{A}(\mathsf{CRS})) = \mathsf{corrupted}\right] &= \Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{sender}, \boldsymbol{\sigma}) \text{ s.t. } \Sigma_b = f(\boldsymbol{\sigma})\right] \\ &\quad + \Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})\right]\end{aligned}$$

Since the case of corrupted receiver is identical to the one in Construction 6.4, then we derive that the first term is negligible by Claim 6.6. The following claim concludes the proof:

Claim 7.8. *For every $b \in \{0,1\}$ there exists a negligible function $\mu_b(\cdot)$ such that:*

$$\Pr\left[\mathcal{A}(\mathsf{CRS}) = (b, \mathsf{receiver}, \mathsf{ch}) \text{ s.t. } \mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})\right] \leq \mu_b(\lambda) \ .$$

Proof. Assume without loss of contradiction that $b = 0$. We prove the claim via hybrid argument.

- $\mathbf{H}_1$: In this experiment, the adversary receives CRS as generated by GenCRS and the output of the experiment is 1 if it outputs $(b, \mathsf{receiver}, \mathsf{ch})$ for which $\mathsf{CHR}_\mathsf{b} = \mathsf{f}(\mathsf{ch})$.
- $\mathbf{H}_2$: In this experiment, we change the CRS that the adversary receives. In $\pi^{(R)}$ we use the witness $((\bot, \bot, \bot), (\mathsf{ch}_1, \mathbf{S}_1, \mathbf{S}'_1))$. The output of the experiment is defined to be the same as in $\mathbf{H}_1$.
- $\mathbf{H}_3$: Like the previous experiment but for which we change each $\beta_{b,j,m,\ell_0,\ell_1}$ to be $\beta_{b,j,m,\ell_0,\ell_1} = \mathsf{OT}^{bit}(0, (\ell_0, \ell_1))$.
- $\mathbf{H}_4$: Like the previous experiment, but for which we change each γ_{b,j,m_0,m_1} to be $\mathsf{OT}^{bit}(0, m_0, m_1))$.

Indistinguishability of hybrids: We claim that the view of the adversary between the different hybrids is indistinguishable. $\mathbf{H}_1$ and $\mathbf{H}_2$ is computationally-indistinguishable from the security of the witness-indistinguishability proof system. $\mathbf{H}_2$ and $\mathbf{H}_3$ are indistinguishable from the security of the sender in OT, i.e., since the input of the receiver is always 0 in all those transcript, the adversary cannot see the information in the second message. The same argument also applies to claim the indistinguishability of $\mathbf{H}_3$ and $\mathbf{H}_4$.

Finally, in $\mathbf{H}_4$, the adversary receives CRS that contains no information about ch_b, except for giving $f(\mathsf{ch}_b)$. Therefore, the probability that $\mathcal{A}$ outputs ch_b in $\mathbf{H}_4$ is negligible, as guaranteed from the security of the one-way function. □

□

Procedure 7.9: Extracting Transcript for Receiver's Accountability

Input: CRS^*, a branch $b \in \{0,1\}$, $\mathbf{y} = (y_1, \ldots, y_\ell)$ as the input of the receiver, randomness $\mathbf{r} = (\mathbf{r}[1], \ldots, \mathbf{r}[\ell])$ for the receiver, $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_n)$ as the shift of the sender.

Goal: Generating transcript corresponding to input Δ_i everywhere except for the first λ positions i for which $y_i = 0$, in which the input of the sender is $\Delta_i \oplus \mathsf{ch}_b[j]$, where ch_b is the challenge in the CRS.

The procedure:

1. Compute $\left(\mathsf{GC}, \left\{K^{(x)}_{i,0}, K^{(x)}_{i,1}, K^{(y)}_{i,0}, K^{(y)}_{i,1}\right\}_{i\in[\ell]}\right) \leftarrow \mathsf{Garble.Gen}\left(1^\lambda, C; \mathsf{r}_{\mathsf{Garble}}\right)$.
2. Let $\lambda = |K^{(x)}_{i,0}|$ be the length of each one of the keys.

3. Compute the transcript as follows. Set $j = 1$. For every $i \in [\ell]$ generate $\mathsf{msg}_S[i]$ as follows:
   If $y_i = 1$ or $j > \lambda$: then generate $\mathsf{msg}_R[i]' = \mathsf{OT}_1^{2\lambda+1}(\mathsf{CRS}_{\mathsf{OT}}, 1; R_i)$ using some randomness R_i. Secret share $sh_i^0 \oplus sh_i^1 = \Delta_i$, and set

$$\mathsf{msg}_S[i]' \;=\; \mathsf{OT}_2^{2\lambda+1}\left(\mathsf{CRS}_{\mathsf{OT}},\ \mathsf{msg}_R[i],\ \left(K_{i,0}^{(x)} \| sh_i^0 \| K_{i,0}^{(y)},\quad K_{i,\Delta_i}^{(x)} \| sh_i^1 \| K_{i,1}^{(y)}\right)\right)$$

   Then, use $(\mathsf{msg}_R[i]', \mathsf{msg}_S[i]') \leftarrow \mathsf{Rerand}_{\mathsf{OT}}((\mathsf{msg}_R[i]', \mathsf{msg}_S[i]'))$ to rerandomize this string OT transcript such that the randomness of the receiver is $\mathbf{r}[i]$, and assign the result in $\mathsf{trans}[i]$.

   If $y_i = 0$ and $j < \lambda$: then the goal is to produce a transcript $\mathsf{trans}[i]$ for string OT in which the input of the receiver is 0 and its randomness is $\mathbf{r}[i]$, and the transcript correspond to:

$$\mathsf{trans}[i] = \mathsf{OT}_2^{2\lambda+1}\left(\mathsf{CRS}_{\mathsf{OT}}, \mathsf{OT}_1^{2\lambda+1}(\mathsf{CRS}_{\mathsf{OT}}, 0; \mathbf{r}[i]),\quad K_{i,0}^{(x)} \| sh_i^0 \| K_{i,0}^{(y)},\quad K_{i,\mathsf{ch}_b[j]\oplus\Delta_i}^{(x)} \| sh_i^1 \| K_{i,1}^{(y)}\right) .$$

   This is accomplished by composing the OTs bit-by-bit as follows:
   (a) **The first $\boldsymbol{\lambda}$ bits:** For every $k \in [\lambda]$: Let $\ell_0 = K_{i,0}^{(x)}[k]$ and $\ell_1 = K_{i,1}^{(x)}[k]$. Set $\mathsf{trans}[i][k] = \beta_{b,j,\Delta_i,\ell_0,\ell_1}$. Note that

$$\mathsf{trans}[i][k] = \mathsf{OT}^{bit}(0, (K_{i,0}^{(x)}[k],\ K_{i,\Delta_i\oplus\mathsf{ch}_b[j]}[k]))$$

   (b) **The $(\boldsymbol{\lambda}+\mathbf{1})$st bit:** Choose $sh_0 \leftarrow \{0,1\}$, and set $m_1 = \Delta_i \oplus sh_0$. Set $\mathsf{trans}[i][\lambda+1] = \gamma_{b,j,sh_0,m_1}$. Note that it holds that

$$\mathsf{trans}[i][\lambda+1] = \mathsf{OT}^{bit}(0, (sh_0, sh_1))$$

   for which $sh_0 \oplus sh_1 = \mathsf{ch}_b[j] \oplus \Delta_i$.
   (c) **The last $\boldsymbol{\lambda}$ bits:** Here we do not need extra information from the CRS. Generate

$$\mathsf{trans}[i][\lambda+2, \ldots, 2\lambda+1] = \mathsf{OT}^{\lambda}(\mathsf{CRS}_{\mathsf{OT}}, (0; R_b), (K_{i,0}^{(y)}, K_{i,1}^{(1)})) .$$

   (d) Overall, $\mathsf{trans}[i]$ correspond to the following string OT:

$$\mathsf{trans}[i] = \mathsf{OT}^{2\lambda+1}\left(\mathsf{CRS}_{\mathsf{OT}}, (0; R_b),\quad K_{i,0}^{(x)} \| sh_i^0 \| K_{i,0}^{(y)},\quad K_{i,\mathsf{ch}_b[j]\oplus\Delta_i}^{(x)} \| sh_i^1 \| K_{i,1}^{(y)}\right) .$$

   We set $\mathsf{trans}[i] = \mathsf{Rerand}_{\mathsf{OT}}(\mathsf{trans}[i])$ such that the randomness of the receiver is $\mathbf{r}[i]$. Increment j.
4. The resulting trans defines $(\mathsf{msg}_R, \mathsf{msg}_S)$.
5. **Output:** $(\mathsf{GC}, \mathsf{msg}_R, \mathsf{msg}_S)$.

Acknowledgments

The authors thank the anonymous reviewers of EUROCRYPT 2021 for many helpful comments.

Gilad Asharov is sponsored by the Israel Science Foundation (grant No. 2439/20), and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234. Hila Dahari is a fellow of the Ariane de Rothschild Women Doctoral Program and supported in part by grants from the Israel Science Foundation (No. 950/15 and 2686/20) and by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness. Vipul Goyal is supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

References

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In *Annual International Cryptology Conference*, pages 468–499. Springer, 2017.

[ADKL19] Prabhanjan Ananth, Apoorvaa Deshpande, Yael Tauman Kalai, and Anna Lysyanskaya. Fully homomorphic NIZK and NIWI proofs. In *Theory of Cryptography TCC 2019*, volume 11892, pages 356–385. Springer, 2019.

[AO12] Gilad Asharov and Claudio Orlandi. Calling out cheaters: Covert security with public verifiability. In *Advances in Cryptology - ASIACRYPT 2012*, volume 7658, pages 681–698. Springer, 2012.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. *IACR Cryptol. ePrint Arch.*, 2020:1024, 2020.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Symposium on Theory of Computing (STOC)*, pages 103–112. ACM, 1988.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. Nizks with an untrusted CRS: security in the face of parameter subversion. In *Advances in Cryptology - ASIACRYPT 2016*, volume 10032, pages 777–804, 2016.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *CRYPTO*, pages 1–18. Springer, 2001.

[BGJ+18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal mpc. In *Annual International Cryptology Conference*, pages 459–487. Springer, 2018.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *Theory of Cryptography Conference*, pages 645–677. Springer, 2017.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 500–532. Springer, 2018.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In *Advances in Cryptology - CRYPTO 2003*, volume 2729, pages 299–315. Springer, 2003.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, 2000.

[CCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In *Theory of Cryptography - TCC '20*, 2020. to appear.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Annual International Cryptology Conference*, pages 19–40. Springer, 2001.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *Advances in Cryptology - CRYPTO '94*, volume 839, pages 257–270. Springer, 1994.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 68–86. Springer, 2003.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *ACM symposium on Theory of computing (STOC)*, pages 494–503, 2002.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *Foundations of Computer Science, FOCS 2000*, pages 283–293, 2000.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge snarks. In *Public-Key Cryptography - PKC 2018*, volume 10769, pages 315–347. Springer, 2018.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[GGJS11] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Bringing people of different beliefs together to do UC. In *Theory of Cryptography - TCC 2011*, volume 6597, pages 311–328. Springer, 2011.

[GK08] Vipul Goyal and Jonathan Katz. Universally composable multi-party computation with an unreliable common reference string. In *Theory of Cryptography, TCC 2008*, volume 4948, pages 142–154. Springer, 2008.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptol.*, 7(1):1–32, 1994.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *CRYPTO 2007*, volume 4622, pages 323–341. Springer, 2007.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In *CRYPTO 2006*, volume 4117, pages 97–111. Springer, 2006.

[Goy07] Vipul Goyal. Reducing trust in the PKG in identity based cryptosystems. In *CRYPTO 2007*, volume 4622, pages 430–447. Springer, 2007.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 736–749, 2021.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *Theory and Applications of Cryptographic Techniques (TCC)*, pages 468–499, 2018.

[HHPV18] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. In *Annual International Cryptology Conference*, pages 488–520. Springer, 2018.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Annual International Cryptology Conference*, pages 335–354. Springer, 2004.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *Advances in Cryptology - EUROCRYPT 2016*, volume 9666, pages 735–763. Springer, 2016.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656, pages 160–176. Springer, 2003.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008*, volume 5157, pages 554–571. Springer, 2008.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–156. Springer, 2021.