Non-Black-Box Approach to Secure Two-Party Computation in Three Rounds

Akshayaram Srinivasan

Tata Institute of Fundamental Research

Abstract. The round complexity of secure two-party computation is a long studied problem with matching upper and lower bounds for the case of black-box simulators (i.e., the simulators that use the adversary as a black-box). In this work, we focus on going beyond this black-box barrier via non-black-box techniques. Specifically, based on standard cryptographic assumptions, we give a construction of a 3-round two-party computation protocol for computing inputless functionalities (such as coin-tossing) that satisfies standard security against malicious senders and ε-security against malicious receivers. Prior to our work such protocols were only known for the case of (weak) zero-knowledge.

1 Introduction

Secure computation [Yao86, GMW87] allows a set of mutually distrusting parties to compute a joint function of their private inputs while providing protection against an arbitrary collusion of corrupted parties who might try to learn additional information about the inputs of honest parties, or try to disrupt the correct computation of the functionality. One of the key research directions in the area of secure computation is to construct protocols that have minimal round complexity. For the special case of two-parties (which is the focus of this work), we have matching upper and lower bounds when simulator is restricted to use the adversary as a black-box. Specifically, Katz and Ostrovsky [KO04] showed that it is not possible to go beyond four rounds (when one of the parties receive the output) and round-optimal constructions for general two-party functionalities are known from any four-round Oblivious Transfer [IPS08, IKO+11] (which is the minimal assumption).

Going beyond this black-box barrier is a fascinating problem with some recent exciting progress. Ananth and Jain [AJ17] gave a construction of a three-round two-party computation protocol with security against adversaries with a priori bounded non-uniform advice. Bitansky, Kalai, and Paneth [BKP18] gave a construction of a 3-round zero-knowledge protocol assuming the (non-standard) assumption of keyless multi-collision resistant hash functions. In a more recent work, Bitansky, Khurana, and Paneth [BKP19] gave a construction of a three-round weak zero-knowledge protocol (which is explained in the next subsection) under standard cryptographic hardness assumptions.

Our Focus. In this work, we focus on developing new techniques that allow us to construct three-round secure computation protocols for an interesting class of functionalities based on standard cryptographic hardness assumptions.

1.1 Our Results

Our main result is a construction of a three-round secure two-party computation protocol for inputless functionalities that satisfy standard security against malicious senders and ε-security against malicious receivers. By ε-security, we mean that for every adversary that is corrupting the receiver and for any non-negligible ε, there exists an ideal world simulator such that the adversary cannot distinguish whether it is interacting with the honest sender or the ideal world simulator except with ε advantage. By inputless functionalities, we mean those functions which do not take private inputs from the parties. Such functionalities could be generically used for sampling from some pre-defined distribution such as coin-tossing, or sampling a common reference string for cryptographic protocols etc. The main theorem we prove in this work is the following:

Theorem 1. *Assuming the existence of a circuit-private Fully Homomorphic Encryption (FHE) scheme*[1]*, hardness of LWE, and either the DLIN or the SXDH assumption, there is a construction of a three-round secure two-party computation protocol for inputless functionalities that satisfy standard security against malicious senders and ε-security against malicious receivers.*

Key Tool. The key technical tool that allows us to prove the above theorem is a 3-message delayed-input weak zero-knowledge protocol. Recall that in a weak zero-knowledge protocol [DNRS99], the zero-knowledge simulator is allowed to depend on both the malicious verifier as well as the distinguisher. More specifically, weak zero-knowledge property guarantees that for any malicious verifier V^*, a distinguisher D and non-negligible distinguishing parameter ε, there is a simulator that can produce a view of the malicious verifier in such a way that D cannot distinguish this view from the real view except with advantage ε. We say that a weak zero-knowledge protocol satisfies delayed-input property if the statement to be proven is only known to the prover before it sends its final round message. Bitansky, Khurana, and Paneth [BKP19] constructed a three-round weak zero-knowledge protocol but unfortunately, this protocol is not delayed-input. In this work, we give a construction of a 3-message weak zero-knowledge protocol that satisfies delayed-input property and we believe this might be of independent interest. Specifically,

Theorem 2. *Assuming the existence of a circuit-private Fully Homomorphic Encryption (FHE) scheme, hardness of LWE, and either the DLIN or the SXDH assumption, there is a construction of a 3-message delayed-input, weak zero-knowledge protocol.*

1.2 Related Work

Non-Black-Box Techniques. There has been a fascinating line of work, starting from the seminal work of Barak [Bar01] that have led to the development of new non-black-box techniques to overcome the known black-box barriers. Non-Black-Box techniques have been particularly fruitful in constructing concurrent secure computation protocols in the plain model [PR03, Pas04, BS05, PR05, Goy13, CLP13, CLP15], constructing protocols with strict polynomial-time simulators [BL02], as well as constructing protocols that are secure against resetting attacks [BGGL01, GS09, DGS09, GM11, BP12, BP13].

Achieving Weaker Security. There is an interesting line of work that have overcome the black-box barrier by considering weaker security guarantees such as super-polynomial time simulation security [BGJ$^+$17, ABG$^+$21]. The work of Badrinarayanan et al. [BGJ$^+$18] has shown the black-box lower bound of Katz-Ostrovsky [KO04] does not hold for a weaker notion of coin tossing which they term as list coin tossing. Intuitively, list coin tossing provides a weaker security guarantee since the simulator is allowed to query the ideal functionality multiple times and forces one of these outputs to the corrupted receiver.

1.3 Open Directions

Our work opens up several interesting research directions and we list a few of them below.

Going Beyond Inputless Functionalities. As we explain in the next section, our techniques seem to only give secure 2PC protocols for computing inputless functionalities. Can we extend these techniques so as to enable computation of general functions?

Extending the Black-Box Lower Bound to ε-Security. The work of Katz-Ostrovsky [KO04] ruled out a construction of standard secure coin-tossing protocol in three rounds for the case of black-box simulators.

[1] We note that such a FHE scheme can be constructed from any (circular-secure) Somewhat Homomorphic Encryption (see [DS16]) which can in turn be instantiated from circular-secure LWE assumption.

It is straightforward to extend their negative result to the case of ε-security for the class of simulators that are allowed to run in (expected) time which is $\mathsf{poly}(1/\varepsilon)$ but make a fixed polynomial number of oracle queries to the adversary. Intuitively, making more queries doesn't seem to help as an adversarial receiver is generating only a single message in the protocol. However, formalizing this intuition seems tricky and is left as an interesting open problem.

Extending to the case of Multiparty Functionalities. Another interesting open direction is to extend our results for computing either general or even specific inputless multiparty functionalities (such as multi-party coin-tossing). To go beyond the two-party setting, one has to deal with non-malleability issues which seems to require new techniques.

2 Technical Overview

In this section, we give a brief overview of the main technical ideas used in our construction of three-round protocol for computing inputless functionalities.

Starting Point. The starting point of our work is the following folklore recipe of constructing secure two-party computation protocols. Take any two-message SFE protocol that is secure against malicious receivers but only has semi-malicious security against senders. Now, attach a zero-knowledge proof to show that the sender's message is well-formed. This would hopefully lead to a secure two-party protocol that is secure against malicious receivers as well as malicious senders. However, making this approach work in the three-round setting is significantly hard and we face the following barriers.

- **3-Round Zero-Knowledge Protocol.** As our focus in on constructing a three-round SFE protocol, we need a three-round zero-knowledge protocol to show the correctness of the sender's SFE message. However, the task of constructing a 3-round zero-knowledge protocol based on standard cryptographic assumptions is notoriously hard and has been open despite significant efforts. In a recent exciting work, Bitansky, Khurana, and Paneth [BKP19] gave a construction of a 3-round weak zero-knowledge protocol based on well-studied cryptographic assumptions.
- **Delayed-Input Property.** We could hope to directly plug-in the above weak zero-knowledge protocol and obtain a construction of three-round SFE protocol that is ε-secure against malicious receivers. However, for this approach to work, we need the weak zero-knowledge to be delayed-input. This is because the statement that needs to proved is only known to the sender before sending its final round message as it corresponds to the correctness of the sender SFE message. As we explain later, the weak zero-knowledge protocol of Bitansky et al. does not satisfy this property and constructing a three-round weak zero-knowledge protocol that is delayed-input encounters significant technical barriers.
- **Extracting the Effective Input of the Malicious Receiver.** Perhaps the most significant challenge that we need to deal with is in extracting the effective input of the malicious receiver. Recall that in the three-round setting, the receiver only sends a single message in the protocol and we need to somehow extract the effective input used by the receiver. This doesn't seem possible if we only use the receiver as a black-box and hence, we need to develop non-black-box techniques for achieving the same.

2.1 Delayed-Input Weak Zero-Knowledge

Starting Point. The starting point of our construction is the recent work of Bitansky, Khurana, and Paneth [BKP19] who gave a three-message weak zero-knowledge protocol based on standard polynomial hardness assumptions (henceforth, denoted as the BKP protocol). Unfortunately, as we explain later, this protocol does not satisfy delayed-input property. We then explain how to modify this construction so that it satisfies this property.

High-Level Overview of the BKP Protocol. The BKP protocol is built on the FLS paradigm [FLS90] with a special *homomorphic trapdoor*. At a high-level, the code of the underlying malicious verifier V^* and the distinguisher D is used to construct a trapdoor simulation circuit HS. This circuit is homomorphically evaluated on the verifier's message to compute the homomorphic trapdoor which is then used by the simulator to fake the honest prover's message. This homomorphic trapdoor is carefully designed such that a malicious prover cannot compute this efficiently but given the code of the malicious verifier V^* and the distinguisher D, the weak zero-knowledge simulator is able to extract this in polynomial time. To explain this idea more concretely, let us first consider a version of the BKP protocol that is secure against *explainable verifiers* [BKP19]. Explainable verifers are a weaker class of malicious verifiers whose messages are in the support of the honest verifier's message distribution. [BKP19] gave a round-preserving compiler from weak zero-knowledge against explainable verifiers to weak zero-knowledge against arbitrary malicious verifiers. As we will see below, the BKP protocol against explaiable verifiers satisfies delayed-input property. However, their compiler that upgrades the security does not preserve this property.

BKP Protocol against Explainable Verifiers. We give a sketch of the BKP protocol for proving statements in the NP language $\mathcal{L}$ against explainable verifiers in Figure 1. The construction uses the following building blocks:

- A non-interactive commitment scheme Com.
- A dense public-key encryption $(\mathsf{Den.Gen}, \mathsf{Den.Enc}, \mathsf{Den.Dec})$. Recall that in a dense encryption scheme, every string that has a same size as that of a valid public key has a corresponding secret key.
- A fully homomorphic encryption $(\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$.
- A compute-and-compare obfuscation $\mathcal{O}$ [GKW17, WZ17]. Recall that compute and compare program $\mathbf{CC}[f, u, m]$ takes an input x and evaluates $f(x)$ and checks if it is equal to u. If it is the case, it outputs m and otherwise, outputs $\perp$. The security of compute and compare obfuscation guarantees that the distribution of $\widetilde{\mathbf{CC}} = \mathcal{O}(\mathbf{CC}[f, u, m])$ is computationally indistinguishable to the obfuscation of a dummy circuit that always outputs $\perp$ as long as u has sufficient min-entropy.
- A random self-reducible public-key encryption $(\mathsf{RSR.Gen}, \mathsf{RSR.Enc}, \mathsf{RSR.Dec}, \mathsf{RSR.}\widetilde{\mathsf{Dec}})$. The first three algorithms have the same syntax as that of a standard public-key encryption scheme. The final algorithm $\mathsf{RSR.}\widetilde{\mathsf{Dec}}$ takes as input a ciphertext ct, uses a distinguisher D that can distinguish between encryptions of two different messages with non-negligible advantage ε and outputs the message encrypted inside ct with overwhelming probability. Constructions of this primitive are known from any rerandomizable encryption [GM82, ElG86, Pai99].
- A ZAP $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ for the language $L = L_1 \vee L_2 \vee L_3$ where each L_i consists of instances of the form
$$z = (\mathsf{stmt}, \mathsf{pk}', \mathsf{pk}, \mathsf{com}, \mathsf{ct}, \mathsf{ct}', \mathsf{ct}'_1) \tag{2.1}$$
such that (in the following, we use $\perp$ to denote a special symbol in the message space of $\mathsf{RSR.Enc}$):
  - $z \in L_1$ iff
$$\exists (s_1, s_2) \quad \text{s.t.} \quad \begin{array}{l} \mathsf{ct}'_1 = \mathsf{Den.Enc}(\mathsf{pk}', s_1; s_2) \wedge \\ \mathsf{ct} = \mathsf{RSR.Enc}(\mathsf{pk}, \perp; s_1) \end{array}$$
  - $z \in L_2$ iff
$$\exists (w, s_4) \quad \text{s.t.} \quad \begin{array}{l} (\mathsf{stmt}, w) \in R_{\mathcal{L}} \wedge \\ \mathsf{ct}' = \mathsf{Den.Enc}(\mathsf{pk}', w; s_4) \wedge \end{array}$$
  - $z \in L_3$ iff
$$\exists \rho \quad \text{s.t.} \quad \mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$$

- **Round-1:** In the first round, the prover P does the following:
  1. It samples $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Den.Gen}(1^\lambda)$.
  2. It sends pk' as the first round message.
- **Round-2:** In the second round, the verifier does the following:
  1. It samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RSR.Gen}(1^\lambda)$.
  2. It samples $u \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{ct}_1 \leftarrow \mathsf{RSR.Enc}(\mathsf{pk}, u)$.
  3. It samples $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$.
  4. It samples $\rho \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$. It then computes $\mathsf{fct} \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}, \rho)$.
  5. It computes $\widetilde{\mathbf{CC}} \leftarrow \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}, \cdot), u, \rho])$.
  6. It samples the first round message r of the ZAP protocol uniformly.
  7. It sends $(\mathsf{pk}, \mathsf{com}, \mathsf{fpk}, \mathsf{fct}, \widetilde{\mathbf{CC}}, r)$ as the second round message.
- **Round-3:** In the final round, the prover does the following:
  1. It computes $\mathsf{ct}' := \mathsf{Den.Enc}(\mathsf{pk}', w; s_4)$ where $s_4 \leftarrow \{0,1\}^\lambda$.
  2. It computes $\pi \leftarrow \mathsf{ZAP.Prove}(r, z, (w, s_4))$.
  3. It computes $\mathsf{ct}'_1 \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', 0^\lambda)$ and $\mathsf{ct} = \mathsf{RSR.Enc}(\mathsf{pk}, \mathbf{0})$ where $\mathbf{0}$ is a default input not equal to $\perp$.
  4. It sends $(\mathsf{stmt}, \pi, \mathsf{ct}', \mathsf{ct}'_1, \mathsf{ct})$ as the final round message.
- **Verifier Checks:**
  1. It checks if $\mathsf{ZAP.Verify}(r, z, \pi) = 1$.
  2. It checks if $\mathsf{RSR.Dec}(\mathsf{sk}, \mathsf{ct}) \neq \perp$.

  If both the checks pass, it accepts.

Figure 1: BKP protocol against Explainable Verifiers

Intuition Behind the Weak Zero-Knowledge Property. The soundness of this protocol is argued using standard techniques. We now give the main intuition behind the weak zero-knowledge property. At a high-level, the weak zero-knowledge simulator uses the explainable verifier V' and the distinguisher D to construct a distinguisher D' that can distinguish between $\mathsf{RSR.Enc}$ of two messages, namely, $\mathbf{0}$ and $\perp$ with advantage μ. If $\mu \geq \varepsilon/2$ (where ε is the distinguishing parameter for WZK), then the simulator can use $\mathsf{RSR.}\widetilde{\mathsf{Dec}}$ and D' to decrypt ct and obtain u. It can then use $\mathsf{FHE.Enc}(\mathsf{fpk}, u)$ in conjunction with $\widetilde{\mathbf{CC}}$ to obtain ρ. It can then use ρ as the trapdoor witness and generate the proof π and complete the interaction with the verifier. On the other hand, if $\mu < \varepsilon/2$, then it follows that the explainable verifier V' cannot distinguish between the cases when ct is an encryption of $\perp$ and when it is an encryption of $\mathbf{0}$ except with advantage $\varepsilon/2$. Thus, the simulator can now switch ct to encrypt $\perp$ and use (s_1, s_2) as the trapdoor witness to compute the proof π and complete the interaction with the verifier. The main novelty in this argument is in the design of the distinguisher D'. Specifically, this distinguisher is not constructed in the clear but is evaluated *under the hood* of the FHE scheme. To give a bit more details, the simulator constructs a homomorphic simulation circuit HS that takes ρ as input and generates the proof π using the witness ρ. It generates the view of the verifier V' and runs the distinguisher D on it. Now, if the combination of V' and D can distinguish between the cases when ct is an encryption of $\mathbf{0}$ and an encryption of $\perp$ with advantage more than $\varepsilon/2$, HS uses $\mathsf{RSR.}\widetilde{\mathsf{Dec}}$ to compute u and output it. We now homomorphically evaluate HS (using the FHE evaluation) on fct (which is an encryption of ρ) to obtain a FHE encryption of u. We feed this as input to $\widetilde{\mathbf{CC}}$ to obtain ρ in the clear. This is used as the trapdoor witness to complete the interaction with the verifier.

Upgrading security from Explainable to Malicious. The protocol given in Figure 1 can be easily verified to satisfy the delayed-input property. However, the key challenge is to preserve this property while upgrading the security against explainable verifiers to security against standard malicious verifiers. Indeed, the transformation described in [BKP19] fails to preserve this property. In their transformation, the first round message is augmented with a dense commitment to the witness. The verifier then shows via a ZAP that either the second round message is generated honestly or the first round commitment is a commitment to a non-witness. This modification is sufficient to show weak zero-knowledge against malicious verifiers but in the course, we have lost the delayed-input property. Thus, we need a new transformation that preserves this property.

Why a Natural Attempt Fails? A natural attempt to upgrade security is for the prover to additionally send a random image y of a one-way permutation f in the first round and in the second round, the verifier sends an additional commitment com' and proves via a ZAP that either the second round message is generated correctly or com' contains the pre-image of y. The one-wayness of f intuitively guarantees that the verifier is forced to generate a valid second round message (which follows from the soundness of ZAP) and thus, we can rely on weak zero-knowledge against explainable verifiers. However, the main issue with this approach is that we get stuck when we try to formalize a reduction that uses a cheating verifier to break the one-wayness of f. Specifically, there does not seem to be a way which allows us to efficiently extract the pre-image of y from the commitment generated by the verifier. One way to get around this issue by relying complexity leveraging [Pas03].In this work, we devise a new technique to overcome this problem by only relying on standard polynomial hardness assumptions.

Our Solution. The key insight behind our solution is that the homomorphic trapdoor simulation paradigm in [BKP19] can be used to efficiently extract "information" from a verifier. Indeed, as described earlier, we used this paradigm to extract the trapdoor ρ from an explainable verifier. We now use this paradigm to extract the pre-image of f efficiently.

To give a bit more details, we modify the above protocol as follows. The verifier now samples another set of messages[2] $(\mathsf{pk}_2, \mathsf{com}_2, \mathsf{fpk}_2, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_2)$ and proves via a ZAP that either

1. $(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{fpk}_1, \mathsf{fct}_1, \widetilde{\mathbf{CC}}_1)$ is correctly sampled, or
2. $(\mathsf{pk}_2, \mathsf{com}_2, \mathsf{fpk}_2, \mathsf{fct}_2)$ is correctly sampled and $\widetilde{\mathbf{CC}}_2 := \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x])$ where $f(x) = y$.

It follows from the soundness of ZAP that either $(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{fpk}_1, \mathsf{fct}_1, \widetilde{\mathbf{CC}}_1)$ is correctly sampled or $(\mathsf{pk}_2, \mathsf{com}_2, \mathsf{fpk}_2, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_2)$ is sampled as above. In the former case, we are back to the realm of explainable verifiers and in the later case, we can hope to break the one-wayness of f. Specifically, if we manage to get hold of $\mathsf{FHE.Enc}(\mathsf{fpk}_2, u_2)$, then we can feed it as input to $\widetilde{\mathbf{CC}}_2$ and obtain x which is a valid pre-image of y. This allows us to obtain a reduction that breaks the one-wayness of f. We now show that we can obtain this information using the homomorphic trapdoor paradigm.

Towards this purpose, we modify the ZAP language L (see Equation 2.1) to include another trapdoor branch L_4 which accepts ρ_2 as a witness that attests com_2 is a commitment to 0 (analogous to trapdoor branch L_3 defined earlier). We also modify ct to be a "double-encryption" of the message $\mathbf{0}$ under public keys pk_1 and pk_2. In the case where only $(\mathsf{pk}_2, \mathsf{com}_2, \mathsf{fpk}_2, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_2)$ is correctly generated, we construct an homormorphic simulation trapdoor HS_2 (analogous to HS described earlier) and run it on fct_2. This outputs an FHE encryption of u_2 and we use this to extract x from $\widetilde{\mathbf{CC}}_2$. This allows us to contradict the one-wayness of f. However, recall that HS_2 (resp., HS_1) is guaranteed to output x (resp., ρ_1) if the verifer/distinguisher pair is able to distinguish ct being an encryption of $\mathbf{0}$ from an encryption of $\perp$ with non-negligible advantage. Thus, to conclude the argument, we show that if we are unable to extract ρ_1 from $\widetilde{\mathbf{CC}}_1$ and x from $\widetilde{\mathbf{CC}}_2$, then the malicious verifier V^* and the distinguisher D is unable to distinguish between RSR encryptions of $\mathbf{0}$ and $\perp$ except with probability $O(\varepsilon)$ (this argument in formalized in Lemma 11). In this case, we use the trapdoor witness (s_1, s_2) (for the trapdoor branch L_1) to compute the proof π. This allows us to prove the weak zero-knowledge of the protocol that additionally satisfies delayed-input property.

A Subtle Non-Malleability Issue in Proving Soundness. While attempting to prove the soundness of the above described protocol, we encounter a subtle non-malleability issue. A natural strategy to prove the soundness is to fix the first round message from the prover non-uniformly and extract the one-way

[2] We use subscript 1 to denote the original second round message of the verifier in Figure 1.

permutation pre-image of y. In a sequence of hybrids, we switch $(\mathsf{pk}_2, \mathsf{com}_2, \mathsf{fpk}_2, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_2)$ to the correct distribution (as described in Point-2 above) and then use the witness indistinguishability property of the ZAP protocol to use this witness instead of the witness for correct generation of $(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{fpk}_1, \mathsf{fct}_1, \widetilde{\mathbf{CC}}_1)$. Once we have done this, we can use the soundness of the protocol against explainable verifiers to complete the argument. However, this strategy encounters the following roadblock. Specifically, when we switch com_2 from a commitment of 1 to a commitment of 0, we inadvertently "activate" the trapdoor branch L_4. Thus, when we make this switch, the cheating prover could start using the witness for the trapdoor branch L_4 and there is no way for us to detect this. Hence, we cannot reduce the soundness of the overall protocol to the soundness of the protocol against explainable verifiers.

To overcome this non-malleability issue, we add an additional ciphertext ct'_2 in the third round message sent by the prover and modify the trapdoor branch L_4 to show that this ciphertext ct'_2 is a valid encryption of ρ_2 under pk' and ρ_2 attests that com_2 is a commitment to 0. With this modification, we can use the (non-uniformly fixed) secret key sk' of the public key pk' to decrypt ct'_2 and check if ρ_2 is a valid randomness for a commitment to 0. From the perfect binding property of the commitment, such a ρ_2 cannot exist when com_2 is a commitment to 1. From the hiding property of the commitment, it follows that when we switch com_2 from a commitment to 1 to a commitment to 0, the prover cannot generate ct'_2 that encrypts a valid opening to 0. This allows us to prove the soundness of the protocol. However, as explained below, this introduces new issues in proving the weak zero-knowledge property which we explain next.

Proving the Weak Zero-Knowledge. Recall that HS_2 was the homomorphic trapdoor simulation circuit that we designed to extract the pre-image x of y. This circuit used the trapdoor witness ρ_2 (which is given as a FHE encryption fct_2 under fpk_2) to generate the proof π. However, with the above modification that helped in proving the soundness, we additionally need to generate an encryption of ρ_2 under pk'. A natural way to obtain this (under the hood of the FHE) is to run $\mathsf{FHE.Eval}$ on fct_2 for computing the functionality $\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s)$ (where s is uniformly chosen). This gives an $\mathsf{FHE.Enc}$ of ct'_2 under fpk_2 and we can use this to homomorphically evaluate HS_2. However, in the real protocol execution, ct'_2 is generated as an encryption of some default value (say, the all zeroes string) whereas in the modified execution, it is generated as an encryption of ρ_2. Intuitively, these two executions should be computationally indistinguishable from the semantic security of $\mathsf{Den.Enc}$. However, proving this involves many subtleties.

Firstly, we need the FHE scheme to be circuit-private so that information about the randomness s used in generating ct'_2 is not leaked. Secondly, to reduce the indistinguishability to the semantic security of $\mathsf{Den.Enc}$, we need the value ρ_2 in the clear (this is needed for the interaction with the challenger for $\mathsf{Den.Enc}$). However, ρ_2 is only available as an FHE encryption under fpk_2 and unless we break open the FHE encryption by running in super-polynomial time, we cannot hope to obtain ρ_2 in the clear. This seems to require sub-exponential hardness assumptions which we want to avoid.

To overcome this conundrum, we make use of the leakage lemma [GW11, JP14, CCL18]. The leakage-lemma states that any "short" inefficiently computable leakage from some distribution X could be efficiently simulated as long as we can tolerate a non-negligible loss in the distinguishing advantage. To use the leakage lemma, we encrypt ρ_2 bit-by-bit and leak one bit of ρ as the inefficient leakage. The leakage lemma guarantees that this inefficient one bit leakage can be efficiently simulated albeit with a small (but non-negligible) loss in the distinguishing advantage. Using this lemma and the security of $\mathsf{Den.Enc}$, we can switch ct'_2 from encryption of all zeroes string to an encryption of ρ_2 one bit at a time. This argument is formalized in Claim 4.4. Once we have made this switch, we can use ρ_2 as the trapdoor witness to design HS_2. This allows us to complete the proof of the weak zero-knowledge property.

The complete description of the delayed-input weak zero-knowledge protocol along with its analysis appears in Section 4.

2.2 Three-Round Two-Party Computation

Coming back to our initial recipe, we could try to directly plug-in the delayed-input weak zero-knowledge protocol with a two-message SFE scheme that is secure against malicious receivers and hope to get a protocol that is ε-secure against malicious receivers. However, this is not as straightforward as it seems and we encounter significant barriers to make this work.

Main Challenge. The main challenge we face here is how to extract the effective input used by the adversarial receiver. Only after we have extracted this input, we could query the ideal functionality on this input and "force" the output obtained from the ideal functionality to the corrupted receiver. Wait! Isn't our SFE protocol already secure against malicious receivers? Unfortunately, these two message SFE protocols in the plain model [NP01, AIR01, Kal05, HK12, BD18, DGI+19] only allow super-polynomial time extraction of the adversarial receiver input and we cannot hope to use these extractors if we want a polynomial time simulator. Since the receiver only sends a single message in the protocol, black-box techniques to extract the receiver's input seem insufficient and hence, we need to develop new non-black-box techniques for the same.

Our Solution. The main idea behind our solution is to make use of the homomorphic trapdoor paradigm to extract the effective receiver's input. Recall that we used this paradigm to extract the pre-image of the one-way permutation, and perhaps, we could use this to extract the effective receiver input in an analogous way. However, one key difference between these two settings is that in the case of two-party computation protocol, the malicious receiver expects to obtain the output of the computed functionality if it sends a valid second round message. This must be contrasted with the zero-knowledge setting where the malicious verifier only gets an accepting transcript. This creates the following circularity issue. In order to extract the effective receiver input, we need to homomorphically evaluate the homomorphic trapdoor simulation circuit HS. However, this circuit needs to generate a third-round message which delivers the output of the functionality to the malicious receiver. This in particular, means that we must have somehow extracted the effective input of the receiver before this and hence, the circularity. To break this circularity, we only consider securely computing inputless functionalities. Specifically, inside the homomorphic simulation circuit, we could generate a final round SFE protocol using an independently chosen random input on behalf of the sender and this is identically distributed to the real execution. Hence, the homomorphic trapdoor simulation succeeds in extracting the effective receiver input which we could use to force an output provided by the ideal functionality.

Problem with Weak Extraction. A subtle point to note here is that this trapdoor simulation paradigm only guarantees "weak-extraction," meaning that only if the adversary is able to distinguish between the two RSR encryptions with non-negligible advantage, we can extract the message. Thus, to be compatible with this "weak-extraction" guarantee, we encrypt the second round SFE message under the RSR encryption. Specifically, if the adversary is unable to distinguish, we switch this RSR encryption to an encryption of some junk value. In that case, the adversary does not obtain the output of the SFE functionality. On the other hand, if the adversary is able to distinguish, then the "weak-extraction" allows us to extract the effective receiver input using the homomorphic trapdoor simulation paradigm.

The full description of the construction of the three-round two-party secure computation protocol appears in Section 5.

Organization. In Section 4, we give our construction of delayed-input weak zero-knowledge protocol. In Section 5, we give our construction of three-round secure two-party computation protocol for inputless functionalities.

3 Preliminaries

Let λ denote the cryptographic security parameter. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$ there exists λ_0 such that for all $\lambda > \lambda_0$ we have $\mu(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$. A function that is not negligible is called a non-negligible function. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function. For any $i \in [n]$, let x_i denote the symbol at the

i-th co-ordinate of x, and for any $T \subseteq [n]$, let $x_T \in \{0,1\}^{|T|}$ denote the projection of x to the co-ordinates indexed by T. We use $\mathsf{supp}(X)$ to denote the support of a random variable X.

For a probabilistic algorithm A, we denote $A(x;r)$ to be the output of A on input x with the content of the random tape being r. When r is omitted, $A(x)$ denotes a distribution. For a finite set S, we denote $x \leftarrow S$ as the process of sampling x uniformly from the set S. We will use PPT to denote Probabilistic Polynomial Time algorithm. Unless it is clear from context, we assume w.l.o.g. that the length of the randomness for all cryptographic algorithms is λ.

We say that two distribution ensembles $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable if for every non-uniform PPT distinguisher D, we have $|\Pr[D(1^\lambda, X_\lambda) = 1]| - \Pr[D(1^\lambda, Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

3.1 Trapdoor Generation Protocol

We consider a three-round trapdoor generation protocol $(\mathsf{TD}_1, \mathsf{TD}_2, \mathsf{TD}_3, \mathsf{TDVerify})$ from the work of [BGJ+18] (based on a digital signature scheme) that satisfies the following properties:

- Given any first round message td_1 from the malicious sender, there is a trapdoor x such that $\mathsf{TDVerify}(x, \mathsf{td}_1) = 1$.
- **Soundness.** Any malicious adversary corrupting the receiver and interacting with an honest sender cannot output x such that $\mathsf{TDVerify}(x, \mathsf{td}_1) = 1$ except with negligible probability.
- **Extraction.** There exists an (expected) PPT extractor Ext that interacts with any malicious sender and outputs x such that $\mathsf{TDVerify}(x, \mathsf{td}_1) = 1$ with overwhelming probability.

3.2 Dense Public-Key Encryption

We recall the definition of dense public-key encryption.

Definition 1. *A public-key encryption scheme* $(\mathsf{Den.Gen}, \mathsf{Den.Enc}, \mathsf{Den.Dec})$ *is a dense public-key encryption scheme for message space* $\{0,1\}^{p(\lambda)}$ *(for some polynomial* p*) if:*

- ***Correctness:*** *For any* λ *and message* $m \in \{0,1\}^{p(\lambda)}$, *we have:*

$$\Pr[\mathsf{Den.Dec}(\mathsf{sk}, \mathsf{ct}) = m] = 1$$

  where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Den.Gen}(1^\lambda)$, $\mathsf{ct} \leftarrow \mathsf{Den.Enc}(\mathsf{pk}, m)$.
- ***Security:*** *For any two messages* $m_0, m_1 \in \{0,1\}^{p(\lambda)}$, *we have:*

$$\{\mathsf{pk}, \mathsf{Den.Enc}(\mathsf{pk}, m_0)\}_\lambda \approx_c \{\mathsf{pk}, \mathsf{Den.Enc}(\mathsf{pk}, m_1)\}_\lambda$$

  where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Den.Gen}(1^\lambda)$.
- ***Dense Public Keys:*** *For any* λ *and any string* $\mathsf{pk}' \in \{0,1\}^{|\mathsf{pk}|}$ *where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Den.Gen}(1^\lambda)$, *there exists* sk' *such that* $(\mathsf{pk}', \mathsf{sk}') \in \mathsf{Den.Gen}(1^\lambda)$.

Instantiations. Dense public-key encryption schemes can be instantiated based on the DDH/SXDH assumption [ElG86], DLIN assumption [BBS04], or the LWE assumption based on the dual Regev system [Reg05, GPV08].

3.3 Fully-Homomorphic Encryption

In this subsection, we recall the notion of fully homomorphic encryption scheme with statistical circuit privacy. Here, we consider a definition where there is a Setup algorithm that outputs a common random string r and the $\mathsf{FHE.Gen}$ takes as input this r and samples a public-key, secret-key pair. This is done to ensure that a public-key, secret-key pair chosen using bad randomness does not affect the correctness of the decryption.[3]

Syntax. A fully homomorphic encryption (FHE) consists of the following algorithms $(\mathsf{Setup}, \mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval}, \mathsf{Sanitize})$. The Setup algorithm outputs a uniformly chosen random string r and $\mathsf{FHE.Gen}$ takes in r and outputs $(\mathsf{pk}, \mathsf{sk})$ pair. $\mathsf{FHE.Enc}$ and $\mathsf{FHE.Dec}$ have the same syntax as that of any public key encryption scheme. The algorithm $\mathsf{FHE.Eval}$ takes as input public key pk and a description of a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and a ciphertext fct encrypting an n-bit message and outputs a new ciphertext fct'. $\mathsf{Sanitize}$ takes in the public key and a FHE ciphertext fct and outputs another ciphertext fct'.

Definition 2. *A tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval}, \mathsf{Sanitize})$ *is said to be a fully homomorphic secret-key encryption scheme with statistical circuit privacy if:*

- ***Correctness.*** *With probability* $1 - 2^{-\lambda}$ *over the choice of* r *output by* $\mathsf{Setup}(1^\lambda)$*, for any* $x \in \{0,1\}^n$*, for any circuit* $C : \{0,1\}^n \to \{0,1\}^m$*, for any* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{FHE.Gen}(r)$*,* $\mathsf{fct} \in \mathsf{FHE.Enc}(\mathsf{sk}, x)$*,*
$$\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{fct})) = C(x)$$
- ***Security.*** *For any two messages* $x_0, x_1 \in \{0,1\}^n$ *and for any* r *in the support of* $\mathsf{Setup}(1^\lambda)$*, we have:*
$$\{(\mathsf{pk}, \mathsf{FHE.Enc}(\mathsf{pk}, x_0)) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Gen}(r)\}_\lambda \approx_c \{(\mathsf{pk}, \mathsf{FHE.Enc}(\mathsf{pk}, x_1)) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Gen}(r)\}_\lambda$$
- ***Compactness.*** *There exists a fixed polynomial* $\mathsf{poly}(\cdot)$ *such that for any* $r \in \mathsf{Setup}(1^\lambda)$*, for any circuit* $C : \{0,1\}^n \to \{0,1\}^m$*,* $x \in \{0,1\}^n$*, any* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{FHE.Gen}(r)$*, and* $\mathsf{fct} \in \mathsf{FHE.Enc}(\mathsf{sk}, x)$*, we have:*
$$|\mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{fct})| \leq \mathsf{poly}(\lambda, m)$$
- ***Correctness of*** $\mathsf{Sanitize}$***.*** *With probability* $1 - 2^{-\lambda}$ *over the choice of* r *output by* $\mathsf{Setup}(1^\lambda)$ *and for any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Gen}(r)$ *and for all* fct *in the support of ciphertext space, we have:*
$$\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{Sanitize}(\mathsf{pk}, \mathsf{fct})) = \mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{fct})$$
- ***Circuit Privacy.*** *With probability* $1 - 2^{-\lambda}$ *over the choice of* r *output by* $\mathsf{Setup}(1^\lambda)$*, we have for any* $t \in \{0,1\}^*$ *and* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{FHE.Gen}(r; t)$ *and for all* $(\mathsf{fct}, \mathsf{fct}')$ *in the support of ciphertext space such that* $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{fct}) = \mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{fct}')$*:*
$$\{r, t, \mathsf{Sanitize}(\mathsf{pk}, \mathsf{fct})\} \approx_s \{r, t, \mathsf{Sanitize}(\mathsf{pk}, \mathsf{fct}')\}$$

Instantiation. An FHE scheme satisfying Definition 2 is constructed in [DS16] based on the (circular-secure) Somewhat Homomorphic Encryption (SHE) which in turn can be instantiated based on (circular-secure) Learning with Errors [Reg05] assumption. We note that for our applications, it is sufficient if the FHE scheme satisfies computational circuit privacy instead of statistical one.

[3] Specifically, if there is a FHE construction such that for $1/2^\lambda$ fraction of the random coins of $\mathsf{FHE.Gen}$, there is a decryption error. Suppose for a uniformly chosen r output by the Setup algorithm, if we set the random coins of $\mathsf{FHE.Gen}$ to be $r \oplus \mathsf{PRG}(t)$ where $|t| = \lambda/2$. Then, with probability $1 - 2^{-\lambda/2}$ over the choice of r, for any maliciously chosen t, there is no deceryption error.

3.4 Random Self-Reducible Public-Key Encryption

We now recall the notion of random self-reducible public-key encryption [BM82]. This subsection is mostly taken verbatim from [BKP19].

Syntax. An random self-reducible public-key encryption (in short, an RSR encryption) consists of a tuple of PPT algorithms $(\mathsf{RSR.Gen}, \mathsf{RSR.Enc}, \mathsf{RSR.Dec}, \mathsf{RSR.}\widetilde{\mathsf{Dec}})$. The first three algorithms have the standard syntax for a public-key encryption scheme. The final algorithm $\mathsf{RSR.}\widetilde{\mathsf{Dec}}^{\mathsf{D}}(\mathsf{ct}, \mathsf{pk}, 1^{1/\varepsilon})$ which we given as inputs a ciphertext ct, a public key pk and a (distinguishing) parameter $1^{1/\varepsilon}$, and oracle access to a distinguisher D outputs a plaintext message m.

Definition 3. *A public-key encryption scheme* $(\mathsf{RSR.Gen}, \mathsf{RSR.Enc}, \mathsf{RSR.Dec}, \mathsf{RSR.}\widetilde{\mathsf{Dec}})$ *for message space* $\{0,1\}^{p(\lambda)}$ *(for some polynomial* p*) is random self-reducible if:*

- ***Correctness:*** *For any* λ *and message* $m \in \{0,1\}^{p(\lambda)}$*, we have:*

$$\Pr[\mathsf{RSR.Dec}(\mathsf{sk}, \mathsf{ct}) = m] = 1$$

  where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RSR.Gen}(1^\lambda)$*,* $\mathsf{ct} \leftarrow \mathsf{RSR.Enc}(\mathsf{pk}, m)$*.*
- ***Security:*** *For any two messages* $m_0, m_1 \in \{0,1\}^{p(\lambda)}$*, we have:*

$$\{\mathsf{pk}, \mathsf{RSR.Enc}(\mathsf{pk}, m_0)\}_\lambda \approx_c \{\mathsf{pk}, \mathsf{RSR.Enc}(\mathsf{pk}, m_1)\}_\lambda$$

  where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RSR.Gen}(1^\lambda)$*.*
- ***Random Self-Reducibility:*** *For any public key* $\mathsf{pk} \in \mathsf{RSR.Gen}(1^\lambda)$*, it holds that for any (probabilistic) distinguisher* D*, any two messages* $m_0, m_1 \in \{0,1\}^{p(\lambda)}$ *and non-negligible* ε*,*
  if

$$|\Pr[\mathsf{D}(\mathsf{RSR.Enc}(\mathsf{pk}, m_0)) = 1] - \Pr[\mathsf{D}(\mathsf{RSR.Enc}(\mathsf{pk}, m_1)) = 1]| \geq \varepsilon$$

  then, for any $m \in \{0,1\}^{p(\lambda)}$ *and* $\mathsf{ct} \in \mathsf{RSR.Enc}(\mathsf{pk}, m)$*,*

$$\Pr[\mathsf{RSR.}\widetilde{\mathsf{Dec}}^{\mathsf{D}}(\mathsf{ct}, \mathsf{pk}, 1^{1/\varepsilon}) = m] \geq 1 - 2^\lambda$$

  where the probability is over the random coins of $\mathsf{RSR.}\widetilde{\mathsf{Dec}}$ *and* D*.*

Remark 1. In [BKP19], RSR encryption is defined only for the case where the messages are bits. It is straightforward to extend this definition to arbitrary length messages by encrypting bit by bit.

Instantiation. Random self-reducible encryption can be constructed from the DDH/SXDH assumption [ElG86], or based on the DLIN assumption [BBS04]. Bitansky et al. [BKP19] gave a construction of a relaxed notion of RSR encryption from LWE with sub-exponential modulus-to-noise ratio. We note that this relaxed notion is also sufficient for our purposes.

3.5 Compute and Compare Obfuscation

We give the definition of compute and compare obfuscation. This subsection is mostly taken verbatim from [BKP19].

Definition 4 (Compute and Compare Programs). *Let* $f : \{0,1\}^n \to \{0,1\}^\lambda$ *be a circuit, and let* $u \in \{0,1\}^\lambda$ *and* $m \in \{0,1\}^{p(\lambda)}$ *(for some polynomial* p*) be two strings. Then,* $\mathbf{CC}[f, u, m](x)$ *outputs* m *if* $f(x) = u$*, and outputs* $\perp$ *otherwise.*

We now define compute and compare (CC) obfuscators. In what follows $\mathcal{O}$ is a PPT algorithm that takes as input a CC circuit $\mathbf{CC}[f, u, m]$ and outputs a new circuit $\widetilde{\mathbf{CC}}$. (We assume that the CC circuit $\mathbf{CC}[f, u, m]$ is given in some canonical description from which f, u, and m can be read.)

Definition 5. *A PPT algorithm $\mathcal{O}$ is a compute and compare obfuscator if:*

– ***Perfect Correctness:*** *For any $f : \{0,1\}^n \to \{0,1\}^\lambda$, $u \in \{0,1\}^\lambda$, $m \in \{0,1\}^{p(\lambda)}$,*

$$\Pr[\forall x \in \{0,1\}^n, \widetilde{\mathbf{CC}}(x) = \mathbf{CC}[f,u,m](x)] = 1$$

where $\widetilde{\mathbf{CC}} \leftarrow \mathcal{O}(\mathbf{CC}[f,u,m])$.

– ***Simulation:*** *There exists a PPT simulator Sim such that for any $f_\lambda : \{0,1\}^n \to \{0,1\}^\lambda$ and any $m_\lambda \in \{0,1\}^{p(\lambda)}$:*

$$\{\widetilde{\mathbf{CC}} : u \leftarrow \{0,1\}^\lambda, \widetilde{\mathbf{CC}} \leftarrow \mathcal{O}(\mathbf{CC}[f_\lambda, u, m_\lambda])\}_\lambda \approx_c \{\mathsf{Sim}(1^\lambda, 1^{|f_\lambda|}, 1^{|m_\lambda|})\}_\lambda$$

Instantiation. Compute-and-Compare obfuscation can be constructed based on the Learning with Errors assumption [GKW17, WZ17, GKVW20].

3.6 ZAPs

ZAPs [DN00] are two-message public-coin witness indistinguishable proofs. It consists of two PPT algorithms ($\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify}$). $\mathsf{ZAP.Prove}$ takes as input a string $r \in \{0,1\}^{p(\lambda)}$ (for some polynomial p), an instance x of an NP language L and witness w attesting that $x \in L$ and outputs a proof π. $\mathsf{ZAP.Verify}$ takes r, x, and π as inputs and outputs 1/0.

Definition 6. ($\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify}$) *is said to be a ZAP proof system for an* NP *language L (with witness relation R_L) if it satisfies:*

– ***Correctness.*** *For any $x \in L$, any w such that $(x,w) \in R_L$, any string $r \in \{0,1\}^{p(\lambda)}$,*

$$\Pr[\mathsf{ZAP.Verify}(r, x, \mathsf{ZAP.Prove}(r, x, w)) = 1] = 1$$

– ***Soundness.*** *For any cheating (unbounded) prover P^* there exists a negligible function μ,*

$$\Pr[\mathsf{ZAP.Verify}(r, x, \pi) = 1 \wedge x \notin L | r \leftarrow \{0,1\}^{p(\lambda)}, (x, \pi) \leftarrow P^*(r)] \leq \mu(\lambda)$$

– ***Witness Indistinguishability.*** *For any $x \in L$ and witnesses w_0, w_1 such that $R_L(x, w_0) = R_L(x, w_1) = 1$ and for any $r \in \{0,1\}^{p(\lambda)}$, we have:*

$$\{\mathsf{ZAP.Prove}(r, x, w_0)\}_\lambda \approx_c \{\mathsf{ZAP.Prove}(r, x, w_1)\}_\lambda$$

Instantiation. ZAPs can be based on factoring [DN00] or on DLIN/SXDH [GOS12]. It can also be constructed assuming quasi-polynomial hardness of LWE assumption [BFJ+20, GJJM20].

3.7 Leakage Lemma

We now recall the leakage lemma from [GW11, JP14, CCL18].

Theorem 3 ([GW11, JP14, CCL18]). *Let $n, \ell \in \mathbb{N}$, $\varepsilon > 0$ and $\mathcal{D}$ be a family of distinguisher circuits from $\{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$ of size $s = s(n)$. Then, for every distribution (X, Z) over $\{0,1\}^n \times \{0,1\}^\ell$, there exists a simulator $h : \{0,1\}^n \to \{0,1\}^\ell$ such that:*

– *h has size bounded by $s' = O(s 2^\ell \varepsilon^{-2})$.*
– *$(X, Z)$and $(X, h(X))$ are ε-indistinguishable by $\mathcal{D}$.That is for every $D \in \mathcal{D}$,*

$$|\Pr[D(X, Z) = 1] - \Pr[D(X, h(X)) = 1]| \leq \varepsilon$$

3.8 Secure Function Evaluation

A secure function evaluation is a two-message protocol between a sender and a receiver. The receiver on input $x \in \{0,1\}^n$ runs SFE_1 on 1^λ and x to obtain sfe_1 and a secret state st. The sender on input a description of a circuit $C : \{0,1\}^n \to \{0,1\}^m$ runs SFE_2 on sfe_1 and C to obtain sfe_2. The receiver runs out on sfe_2 and the secret state st to obtain a string $y \in \{0,1\}^m$.

Definition 7. *A tuple* $(\mathsf{SFE}_1, \mathsf{SFE}_2, \mathsf{out})$ *is a secure function evaluation protocol if it satisfies:*

– ***Correctness.*** *For any $x \in \{0,1\}^n$ and $C : \{0,1\}^n \to \{0,1\}^m$, we have:*

$$\Pr[\mathsf{out}(\mathsf{sfe}_2, \mathsf{st}) = C(x)] = 1$$

where $(\mathsf{sfe}_1, \mathsf{st}) \leftarrow \mathsf{SFE}_1(1^\lambda, x)$ *and* $\mathsf{sfe}_2 \leftarrow \mathsf{SFE}_2(\mathsf{sfe}_1, C)$*.*

– ***Receiver Message Indistinguishability.*** *For any two inputs $x_0, x_1 \in \{0,1\}^n$, we have:*

$$\{\mathsf{SFE}_1(1^\lambda, x_0)\}_\lambda \approx_c \{\mathsf{SFE}_1(1^\lambda, x_1)\}_\lambda$$

– ***Sender Security.*** *There exists a simulator $\mathsf{Sim}_{\mathsf{SFE}}$ such that for any $x \in \{0,1\}^n$, $r \in \{0,1\}^\lambda$ and $C : \{0,1\}^n \to \{0,1\}^m$, we have:*

$$\{\mathsf{SFE}_2(\mathsf{SFE}_1(1^\lambda, x; r), C)\}_\lambda \approx_c \{\mathsf{Sim}_{\mathsf{SFE}}(1^\lambda, x, r, C(x))\}_\lambda$$

Instantiation. A two-message SFE satisfying the above definition can be constructed from any two-message semi-malicious secure oblivious transfer using the Yao's protocol [Yao86]. Two-message oblivious transfer can be constructed from a variety of assumptions such as DDH/SXDH [NP01], DLIN [LVW20] or LWE [BD18].

4 Delayed-Input Weak Zero-Knowledge

In this section, we give a construction of a delayed-input weak zero-knowledge protocol that runs in three rounds. This is used in the next section to construct a 3-round ε-secure 2PC for inputless functionalities.

4.1 Definition

Syntax. We describe the syntax of a three-round weak zero-knowledge protocol with delayed input property.

– $\mathsf{wZK.P}_1(1^\lambda)$: It is a PPT algorithm run by the prover that takes as input the security parameter in unary and outputs wzk_1.
– $\mathsf{wZK.V}_1(\mathsf{wzk}_1)$: It is a PPT algorithm run by the verifier that takes the first round message wzk_1 generated by the prover and outputs wzk_2 and secret verifier state st_V.
– $\mathsf{wZK.P}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, (x, w))$: It is a PPT algorithm run by the prover that takes the first two messages in the protocol $(\mathsf{wzk}_1, \mathsf{wzk}_2)$, an instance $x \in L$ and a witness $w \in R_L(x)$ and outputs wzk_3.
– $\mathsf{wZK.V}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3, x, \mathsf{st}_V)$: It is a deterministic algorithm run by the verifier that takes the transcript of the protocol $(\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3)$, the instance x and the secret verifier state st_V and outputs 1/0.

Definition 8. *A three-round interactive argument* $(\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2)$ *is a delayed-input, weak zero-knowledge protocol for an* NP *language L (with the witness set $R_L(\cdot)$) if it satisfies:*

– ***Completeness.*** *For any $x \in L$ and $w \in R_L(x)$, we have:*

$$\Pr[\mathsf{wZK.V}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3, x, \mathsf{st}_V) = 1] = 1$$

where $\mathsf{wzk}_1 \leftarrow \mathsf{wZK.P}_1(1^\lambda)$*,* $(\mathsf{wzk}_2, \mathsf{st}_V) \leftarrow \mathsf{wZK.V}_1(\mathsf{wzk}_1)$*,* $\mathsf{wzk}_3 \leftarrow \mathsf{wZK.P}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, (x, w))$*.*

– ***Adaptive Computational Soundness.*** *For any non-uniform (stateful) PPT prover P^*, there exists a negligible function $\mu(\cdot)$ such that:*

$$\Pr\Big[\mathsf{wZK.V}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3, x, \mathsf{st}_V) = 1 \wedge x \notin L \Big| \mathsf{wzk}_1 \leftarrow P^*(1^\lambda),$$
$$(\mathsf{wzk}_2, \mathsf{st}_V) \leftarrow \mathsf{wZK.V}_1(\mathsf{wzk}_1), (\mathsf{wzk}_3, x) \leftarrow P^*(\mathsf{wzk}_2)\Big] \le \mu(\lambda)$$

– ***Weak Zero-Knowledge.*** *For any non-uniform (stateful) PPT malicious verifier V^*, distinguisher D, there exists a (stateful) PPT simulator* $\mathsf{Sim}_{\mathsf{wzk}}$ *such that for any non-negligible error parameter ε and for any instance generator* $\mathsf{InstGen}$*,*

$$|\Pr[\mathsf{REAL}(1^\lambda, V^*, D, \mathsf{InstGen}) = 1] -$$
$$\Pr[\mathsf{IDEAL}(1^\lambda, V^*, D, \mathsf{Sim}_{\mathsf{wzk}}, 1^{1/\varepsilon}, \mathsf{InstGen}) = 1] \le \varepsilon$$

where REAL *and* IDEAL *experiments are described in Figure 2.*

$\mathsf{REAL}(1^\lambda, V^*, D, \mathsf{InstGen})$	$\mathsf{IDEAL}(1^\lambda, V^*, D, \mathsf{Sim}_{\mathsf{wzk}}, 1^{1/\varepsilon}, \mathsf{InstGen})$
– $\mathsf{wzk}_1 \leftarrow \mathsf{wZK.P}_1(1^\lambda)$. – $\mathsf{wzk}_2 \leftarrow V^*(\mathsf{wzk}_1)$ – $(x, w) \leftarrow \mathsf{InstGen}(\mathsf{wzk}_1)$ where $w \in R_L(x)$. – $\mathsf{wzk}_3 \leftarrow \mathsf{wZK.P}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, (x, w))$ – Output $D(\mathsf{wzk}_1, (x, w), \mathsf{wzk}_2, \mathsf{wzk}_3)$.	– $\mathsf{wzk}_1 \leftarrow \mathsf{Sim}_{\mathsf{wzk}}(1^\lambda, 1^{1/\varepsilon})$. – $\mathsf{wzk}_2 \leftarrow V^*(\mathsf{wzk}_1)$ – $(x, w) \leftarrow \mathsf{InstGen}(\mathsf{wzk}_1)$ where $w \in R_L(x)$. – $\mathsf{wzk}_3 \leftarrow \mathsf{Sim}_{\mathsf{wzk}}(\mathsf{wzk}_1, \mathsf{wzk}_2, x, 1^{1/\varepsilon})$ – Output $D(\mathsf{wzk}_1, (x, w), \mathsf{wzk}_2, \mathsf{wzk}_3)$.

Figure 2: Descriptions of REAL and IDEAL.

4.2 Building Blocks

Let $L \in \mathsf{NP}$ be a language with the NP witness relation V_L. The construction uses the following building blocks. The formal definitions can be found in Appendix 3.

– A dense public-key encryption scheme ($\mathsf{Den.Gen}, \mathsf{Den.Enc}, \mathsf{Den.Dec}$). We assume without loss of generality that the encryption is done bit-by-bit.
– A fully-homomorphic encryption scheme ($\mathsf{Setup}, \mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}$, $\mathsf{FHE.Eval}, \mathsf{Sanitize}$) that is statistically circuit-private. We assume without loss of generality that the encryption is done by bit-by-bit.
– A one-way permutation $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$.[4]
– A compute and compare obfuscation $\mathcal{O}$.
– A random self-reducible public-key encryption ($\mathsf{RSR.Gen}, \mathsf{RSR.Enc}, \mathsf{RSR.Dec}, \mathsf{RSR.}\widetilde{\mathsf{Dec}}$).
– A non-interactive commitment scheme Com.
– A ZAP proof system ($\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}}$) for the NP language $\overline{L} = \overline{L}_1 \vee \overline{L}_2$ where $\overline{L}_1$ and $\overline{L}_2$ consists of instances of the form

$$\overline{z} = (y, y', r, \widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, \mathsf{ct}_1, \mathsf{pk}_1, \mathsf{fpk}_1, \mathsf{ct}_2, \mathsf{pk}_2, \mathsf{fpk}_2, \mathsf{fct}_1, \mathsf{fct}_2, \mathsf{com}) \tag{4.1}$$

such that:

[4] We note that the requirement of one-way permutation can be replaced with the DLOG assumption. For the purpose of simplicity of exposition, we go with a one-way permutation.

- $\overline{z} \in \overline{L}_1$ iff

$$\exists(\rho, u_1, r_1, r_2, r_3, r_4, r_5) \qquad \text{s.t.} \qquad \begin{array}{l} (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{RSR.Gen}(1^\lambda; r_1) \wedge \\ \mathsf{ct}_1 = \mathsf{RSR.Enc}(\mathsf{pk}_1, u_1; r_2) \wedge \\ (\mathsf{fpk}_1, \mathsf{fsk}_1) = \mathsf{FHE.Gen}(r; r_3) \wedge \\ \mathsf{fct}_1 := \mathsf{FHE.Enc}(\mathsf{pk}_1, \rho; r_4) \wedge \\ \mathsf{com} = \mathsf{com}(1^\lambda, 0; \rho) \wedge \\ \widetilde{\mathbf{CC}}_1 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_1, \cdot), u_1, \rho]; r_5) \end{array}$$

- $\overline{z} \in \overline{L_2}$ iff

$$\exists(x, x', u_2, r_1, r_2, r_3, r_4, r_5) \qquad \text{s.t.} \qquad \begin{array}{l} y = f(x) \wedge \\ y = f(x') \wedge \\ (\mathsf{pk}_2, \mathsf{sk}_2) = \mathsf{RSR.Gen}(1^\lambda; r_1) \wedge \\ \mathsf{ct}_2 = \mathsf{RSR.Enc}(\mathsf{pk}_2, u_2; r_2) \wedge \\ (\mathsf{fpk}_2, \mathsf{fsk}_2) = \mathsf{FHE.Gen}(r; r_3) \wedge \\ \mathsf{fct}_2 = \mathsf{FHE.Enc}(\mathsf{fpk}_2, x'; r_4) \wedge \\ \widetilde{\mathbf{CC}}_2 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x]; r_5) \end{array}$$

– A ZAP proof ($\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify}$) for the NP language $L = L_1 \vee L_2 \vee L_3 \vee L_4$ where L_1, L_2, L_3, and L_4 consists of instances of the form

$$z = (\mathsf{stmt}, \mathsf{pk}', y', \mathsf{com}, \mathsf{ct}', \mathsf{ct}'_1, \mathsf{ct}'_2, \overline{\mathsf{ct}}) \tag{4.2}$$

such that:

- $z \in L_1$ iff

$$\exists(x', s_1) \qquad \text{s.t.} \qquad \begin{array}{l} y' = f(x') \wedge \\ \mathsf{ct}'_1 = \mathsf{Den.Enc}(\mathsf{pk}', x'; s_1) \wedge \end{array}$$

- $z \in L_2$ iff

$$\exists(s_1, s_2, s_3) \qquad \text{s.t.} \qquad \begin{array}{l} \mathsf{ct}'_2 = \mathsf{Den.Enc}(\mathsf{pk}', (s_1, s_2); s_3) \wedge \\ \overline{\mathsf{ct}} = \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \perp; s_1); s_2) \end{array}$$

- $z \in L_3$ iff

$$\exists(w, s_4) \qquad \text{s.t.} \qquad \begin{array}{l} V_L(\mathsf{stmt}, w) = 1 \wedge \\ \mathsf{ct}' = \mathsf{Den.Enc}(\mathsf{pk}', w; s_4) \wedge \end{array}$$

- $z \in L_4$ iff

$$\exists \rho \qquad \text{s.t.} \qquad \mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$$

4.3 Construction

We give the formal description of the protocol in Figure 3.

4.4 Proof of Security

We now show that the above construction satisfies Definition 8. Completeness is easy to observe.

– $\mathsf{wZK.P}_1(1^\lambda)$: The prover does the following:

1. It samples $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Den.Gen}(1^\lambda)$.
2. It samples a uniform random string $\overline{s} \leftarrow \{0,1\}^{p(\lambda)}$.
3. It samples $x \leftarrow \{0,1\}^\lambda$ and sets $y = f(x)$.
4. It samples $r \leftarrow \mathsf{Setup}(1^\lambda)$.
5. It sends $\mathsf{wzk}_1 = (\mathsf{pk}', r, \overline{s}, y)$ to the verifier.

– $\mathsf{wZK.V}_1(\mathsf{wzk}_1)$: The verifier does the following:

1. It samples $(\mathsf{pk}_1, \mathsf{sk}_1) := \mathsf{RSR.Gen}(1^\lambda; r_1)$ where $r_1 \leftarrow \{0,1\}^\lambda$.
2. It samples $u_1 \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{ct}_1 := \mathsf{RSR.Enc}(\mathsf{pk}_1, u_1; r_2)$ where $r_2 \leftarrow \{0,1\}^\lambda$.
3. It samples $(\mathsf{fpk}_1, \mathsf{fsk}_1) := \mathsf{FHE.Gen}(r; r_3)$ where $r_3 \leftarrow \{0,1\}^\lambda$.
4. It samples $\rho \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$. It then computes $\mathsf{fct}_1 = \mathsf{FHE.Enc}(\mathsf{fpk}_1, \rho; r_4)$ where $r_4 \leftarrow \{0,1\}^\lambda$.
5. It computes $\widetilde{\mathbf{CC}}_1 := \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_1, \cdot), u_1, \rho]; r_5)$ where $r_5 \leftarrow \{0,1\}^\lambda$.
6. It samples $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{RSR.Enc}(1^\lambda)$ and computes $\mathsf{ct}_2 \leftarrow \mathsf{RSR.Enc}(\mathsf{pk}_2, 0^\lambda)$.
7. It samples $(\mathsf{fpk}_2, \mathsf{fsk}_2) \leftarrow \mathsf{FHE.Gen}(r)$. It sets $\mathsf{fct}_2 \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}_2, 0^\lambda)$.
8. It samples $x' \leftarrow \{0,1\}^\lambda$ and computes $y' = f(x')$.
9. It computes $\widetilde{\mathbf{CC}}_2 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot)|}, 1^\lambda)$.
10. It computes $\overline{\pi} \leftarrow \overline{\mathsf{ZAP.Prove}}(\overline{s}, \overline{z}, (\rho, u_1, \{r_i\}_{i\in[5]}))$ (where $\overline{z}$ is described in Equation 4.1).
11. It samples a uniform random string $s \leftarrow \{0,1\}^{p(\lambda)}$.
12. It sends $\mathsf{wzk}_2 = (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{fpk}_1, \mathsf{pk}_2, \mathsf{ct}_2, \mathsf{fpk}_2, y', \mathsf{com}, \mathsf{fct}_1, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, s, \overline{\pi})$ to the prover and sets $\mathsf{st}_V = (sk_1, sk_2, s)$.

– $\mathsf{wZK.P}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, (\mathsf{stmt}, w))$: The prover does the following:

1. It checks if $\overline{\mathsf{ZAP.Verify}}(\overline{s}, \overline{z}, \overline{\pi}) = 1$ and aborts otherwise.
2. It computes $\overline{\mathsf{ct}} := \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \mathbf{0}; s_1); s_2)$ where $s_1, s_2 \leftarrow \{0,1\}^\lambda$ and $\mathbf{0}$ is some default input not equal to $\perp$.
3. It computes $\mathsf{ct}' \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', w; s_4)$ where $s_4 \leftarrow \{0,1\}^\lambda$.
4. It generates $\mathsf{ct}'_1 \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', 0^\lambda)$ and $\mathsf{ct}'_2 \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', 0^{2\lambda})$.
5. It computes $\pi \leftarrow \mathsf{ZAP.Prove}(s, z, (w, s_4))$ (where z is described in Equation 4.2).
6. It sends $\mathsf{wzk}_3 = (\pi, \mathsf{stmt}, \mathsf{ct}'_1, \mathsf{ct}'_2, \overline{\mathsf{ct}}, \mathsf{ct}')$ to the verifier.

– $\mathsf{wZK.V}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3, x, \mathsf{st}_V)$: The verifier does the following checks:

1. It checks if $\mathsf{ZAP.Verify}(s, z, \pi) = 1$.
2. It checks if $\mathsf{RSR.Dec}(\mathsf{sk}_1, \mathsf{RSR.Dec}(\mathsf{sk}_2, \overline{\mathsf{ct}})) \neq \perp$.

If both checks pass, it accepts.

Figure 3: Delayed Input Weak Zero-Knowledge

Proof of Adaptive Computational Soundness Assume for the sake of contradiction that there exists a prover P^* that breaks the adaptive computational soundness property of the protocol with non-negligible probability $\mu(\lambda)$.

We non-uniformly fix the first round message from the prover P^*. Let sk' be the corresponding secret key such that $(\mathsf{pk}', \mathsf{sk}') \in \mathsf{Den.Gen}(1^\lambda)$. Note that by the property of dense encryption scheme such a secret key must exist. Let $x \in \{0,1\}^\lambda$ be such that $y = f(x)$. Let E be the event such that:

1. $\mathsf{ZAP.Verify}(s, z, \pi) = 1$.
2. $w = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}')$ and $(\mathsf{stmt}, w) \notin R_L$.
3. $(s_1, s_2) = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_2)$ and $\overline{\mathsf{ct}} \neq \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \perp; s_1); s_2)$.

We define a sequence of hybrids and let p_i be the probability that E happens in Hyb_i.

– $\underline{\mathsf{Hyb}_0}$: This corresponds to the execution of the protocol with the prover P^*. In Lemma 1, we prove that $p_0 \geq \mu(\lambda)$.

– $\underline{\mathsf{Hyb}_1}$: In this hybrid, we modify the event E to additionally include the condition that $f(\mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}_1')) \neq y'$. In Lemma 2, we rely on the one-wayness of f to argue that $p_1 \geq p_0 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_2}$: In this hybrid, we generate fct_2 as $\mathsf{FHE.Enc}(\mathsf{fpk}_2, x')$. In Lemma 3, we rely on the security of the FHE scheme to to show that $p_2 \geq p_1 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_3}$: In this hybrid, we compute $\widetilde{\mathbf{CC}}_2$ as $\mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x])$ where $u_2 \leftarrow \{0,1\}^\lambda$. In Lemma 4, we rely on the security of the compute and compare obfuscation to show that $p_3 \geq p_2 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_4}$: In this hybrid, we compute ct_2 as $\mathsf{RSR.Enc}(\mathsf{pk}_2, u_2)$. In Lemma 5, we rely on the security of the RSR encryption to show that $p_4 \geq p_3 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_5}$: In this hybrid, we make the following changes:
  1. We compute $(\mathsf{pk}_2, \mathsf{sk}_2) = \mathsf{RSR.Gen}(1^\lambda; r_1)$ where $r_1 \leftarrow \{0,1\}^\lambda$.
  2. We compute $\mathsf{ct}_2 = \mathsf{RSR.Enc}(\mathsf{pk}_2, u_2; r_2)$ where $r_2, u_2 \leftarrow \{0,1\}^\lambda$.
  3. We sample $(\mathsf{fpk}_2, \mathsf{fsk}_2) = \mathsf{FHE.Gen}(1^\lambda; r_3)$
  4. We compute $\mathsf{fct}_2 = \mathsf{FHE.Enc}(\mathsf{fpk}_2, x'; r_4)$
  5. We compute $\widetilde{\mathbf{CC}}_2 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x]; r_5)$
  6. We generate $\overline{\pi} \leftarrow \overline{\mathsf{ZAP.Prove}}(\overline{r}, \overline{z}, (x, x', u_2, r_1, r_2, r_3, r_4, r_5))$.

  In Lemma 6, we rely on the witness indistinguishability of $(\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}})$ to show that $p_5 \geq p_4 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_6}$: In this hybrid, we generate ct_1 as $\mathsf{RSR.Enc}(\mathsf{pk}_1, 0^\lambda)$ instead of $\mathsf{RSR.Enc}(\mathsf{pk}_1, u_1)$. Via a similar argument to Lemma 5, we rely on the security of the RSR encryption to show that $p_6 \geq p_5 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_7}$: In this hybrid, we generate $\widetilde{\mathbf{CC}}_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|\mathsf{FHE.Dec}(\mathsf{fsk}_1, \cdot)|}, 1^\lambda)$. Via a similar argument to Lemma 4, we rely on the security of the compute and compare obfuscation to show that $p_7 \geq p_6 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_8}$: In this hybrid, we generate fct_1 as $\mathsf{FHE.Enc}(\mathsf{fpk}_1, 0^\lambda)$. Via a similar argument to Lemma 3, we rely on the security of FHE encryption to show that $p_8 \geq p_7 - \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_9}$: In this hybrid, we generate com as $\mathsf{Com}(1^\lambda, 1)$. From the hiding property of Com, it follows that $p_9 \geq p_8 - \mathsf{negl}(\lambda)$.

  By the above arguments, it now follows that $p_9 \geq \mu - \mathsf{negl}(\lambda) > 3\mu/4$. In Lemma 7, we rely on the soundness of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ to show that $p_9 \leq \mu/2$ and this is a contradiction.

Lemma 1. $p_0 \geq \mu(\lambda)$.

Proof. Let F be the event that P^* during its interaction with the honest verifier outputs $(\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3, \mathsf{stmt})$ such that $\mathsf{stmt} \notin L$ and honest verifier accepts the transcript $(\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3)$. By assumption, the probability that this event F happens is at least $\mu(\lambda)$. We now show that whenever F happens then E also happens.

Recall that acceptance criterion of the honest verifier involves the following checks:

1. Check if $\mathsf{ZAP.Verify}(s, z, \pi) = 1$.
2. Check if $\mathsf{RSR.Dec}(\mathsf{sk}_1, \mathsf{RSR.Dec}(\mathsf{sk}_2, \overline{\mathsf{ct}})) \neq \perp$.

Therefore, if event F happens then $\mathsf{ZAP.Verify}(s, z, \pi) = 1$. Further, suppose there exists $(s_1, s_2) = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}_2')$ and $\overline{\mathsf{ct}} = \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \perp; s_1); s_2)$ then the second check $\mathsf{RSR.Dec}(\mathsf{sk}_1, \mathsf{RSR.Dec}(\mathsf{sk}_2, \overline{\mathsf{ct}})) \neq \perp$ will fail (from the perfect correctness of the $\mathsf{RSR.Dec}$). Thus, whenever F happens, it follows that $(s_1, s_2) = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}_2')$ and $\overline{\mathsf{ct}} \neq \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \perp; s_1); s_2)$. Finally, since $\mathsf{stmt} \notin L$, it holds that $w = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}')$ is such that $(\mathsf{stmt}, w) \notin R_L$ (this holds for any string w and in particular, holds for $w = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}')$). Therefore, whenever F happens, event E also happens and therefore, $p_0 \geq \mu(\lambda)$.

Lemma 2. *Assuming the one-wayness property of f, we have $p_1 \geq p_0 - \mathsf{negl}(\lambda)$.*

Proof. Assume for the sake of contradiction that there exists a non-negligible function $\varepsilon(\cdot)$ such that $p_1 \leq p_0 - \varepsilon(\lambda)$. Note that the only difference between Hyb_1 and Hyb_0 is that in Hyb_1, we modify the criterion for E to additionally check if $f(\mathsf{Den.Dec}(\mathsf{sk}'.\mathsf{ct}_1')) \neq y'$. Since $p_1 \leq p_0 - \varepsilon(\lambda)$, it now follows that $f(\mathsf{Den.Dec}(\mathsf{sk}'.\mathsf{ct}_1')) = y'$ happens with probability at least $\varepsilon(\lambda)$. We show that this contradicts the one-wayness of f.

We interact with the one-wayness challenger and obtain the challenge y'. We use this to generate the second round message wzk_2 on behalf of the honest verifier. At the end of the interaction with P^*, we check if $f(\mathsf{Den.Dec}(\mathsf{sk}'.\mathsf{ct}'_1)) = y'$ and output x' as the pre-image of y' to the challenger.

Note that since $f(\mathsf{Den.Dec}(\mathsf{sk}'.\mathsf{ct}'_1)) = y'$ happens with probability at least $\varepsilon(\lambda)$, the above reduction breaks the one-wayness property of f with probability $\varepsilon(\lambda)$. This is a contradiction.

Lemma 3. *Assuming the security of the FHE scheme, we have $p_2 \geq p_1 - \mathsf{negl}(\lambda)$.*

Proof. Assume for the sake of contradiction that there exists a non-negligible function $\varepsilon(\cdot)$ such that $p_2 \leq p_1 - \varepsilon(\lambda)$. We now show that this contradicts the security of the FHE encryption scheme.

We interact with the FHE challenger and provide x' and 0^λ as the two challenge messages and r as the output of Setup. We obtain $\mathsf{fct}_2, \mathsf{fpk}_2$ from the challenger. We use it to generate wzk_2 and complete the interaction with the prover P^*. At the end of this interaction, we check if event E happens or not. If E happens, we output 1 and otherwise, output 0.

We note that if fct_2 was generated as an encryption of x' then the view of P^* in the above interaction is identically distributed to Hyb_2. Otherwise, it is identically distributed to Hyb_1. The probability that the above reduction outputs 1 in Hyb_1 is p_1 and the probability that it outputs 1 in Hyb_2 is at most $p_1 - \varepsilon(\lambda)$. Thus, the above reduction breaks the security of the FHE encryption scheme with $\varepsilon(\lambda)$ advantage and this is a contradiction.

Lemma 4. *Assuming the security of the compute and compare obfuscation $\mathcal{O}$, we have $p_3 \geq p_2 - \mathsf{negl}(\lambda)$.*

Proof. Assume for the sake of contradiction that there is a non-negligible function $\varepsilon(\cdot)$ such that $p_3 \leq p_2 - \varepsilon(\lambda)$. We show that this breaks the security of the compute and compare obfuscation.

We interact with the compute and compare obfuscator challenger and provide $\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), x$ to it. The challenger provides $\widetilde{\mathbf{CC}}_2$. We use it to generate wzk_2 on behalf of the honest verifier V^*. At the end of the interaction with prover P^*, we check if event E happens. If it happens, we output 1 and otherwise, we output 0.

We note that if $\widetilde{\mathbf{CC}}_2$ is generated as $\mathsf{Sim}(1^\lambda, 1^{|\mathsf{FHE.Dec}(\mathsf{fsk}_2,\cdot)|}, 1^{|x|})$ then view of P^* in the above interaction is identical to Hyb_2. On the other hand, if $\widetilde{\mathbf{CC}}_2$ was generated as $\mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x])$ for uniformly chosen u_2, then the view of P^* is identically distributed to Hyb_3. The probability that the above reduction outputs 1 in Hyb_2 is p_2 whereas the probability that it outputs 1 in Hyb_2 is at most $p_2 - \varepsilon(\lambda)$. Thus, the above reduction breaks the security of the compute and compare obfuscation with advantage $\varepsilon(\lambda)$ and this is a contradiction.

Lemma 5. *Assuming the security of the RSR encryption scheme, we have $p_4 \geq p_3 - \mathsf{negl}(\lambda)$.*

Proof. Assume for the sake of contradiction that there exists a non-negligible function $\varepsilon(\lambda)$ such that $p_4 \leq p_3 - \varepsilon(\lambda)$. We now show that this contradicts the security of the RSR encryption scheme.

We interact with the RSR encryption challenger and provide uniformly chosen u_2 and 0^λ as the two challenge messages. We obtain $\mathsf{pk}_2, \mathsf{ct}_2$ from the challenger. We generate $\widetilde{\mathbf{CC}}_2$ as $\mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x])$ and generate wzk_2. At the end of the interaction with prover P^*, we check if event E happens. If it happens, we output 1 and otherwise, we output 0.

We note that if ct_2 was generated as encryption of 0^λ then the view of P^* in the above interaction is identical to Hyb_3. Otherwise, it is identical to Hyb_4. The probability that the above reduction outputs 1 in Hyb_3 is p_3 and the probability that it outputs 1 in Hyb_4 is at most $p_3 - \varepsilon(\lambda)$. Thus, the above reduction breaks the security of the RSR encryption scheme with non-negligible advantage $\varepsilon(\lambda)$ and this is a contradiction.

Lemma 6. *Assuming the witness indistinguishability of $(\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}})$, we have $p_5 \geq p_4 - \mathsf{negl}(\lambda)$.*

Proof. Assume for the sake of contradiction that there exists a non-negligible function $\varepsilon(\lambda)$ such that $p_5 \leq p_4 - \varepsilon(\lambda)$. We now show that this contradicts the witness indistinguishability property of $(\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}})$.

We compute $(\mathsf{pk}_2, \mathsf{sk}_2) = \mathsf{RSR.Gen}(1^\lambda; r_1)$ where $r_1 \leftarrow \{0,1\}^\lambda$. We then compute $\mathsf{ct}_2 = \mathsf{RSR.Enc}(\mathsf{pk}_2, u_2; r_2)$ where $r_2, u_2 \leftarrow \{0,1\}^\lambda$. We sample $(\mathsf{fpk}_2, \mathsf{fsk}_2) = \mathsf{FHE.Gen}(1^\lambda; r_3)$. We compute $\mathsf{fct}_2 = \mathsf{FHE.Enc}(\mathsf{fpk}_2, x'; r_4)$. We compute $\widetilde{\mathbf{CC}}_2 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x]; r_5)$ We interact with the external challenger for the witness indistinguishability and provide $\overline{s}$ (present in wzk_1), the instance $\overline{z}$ and the two witnesses as $\overline{w}_1$ which is a witness for $\overline{z} \in \overline{L}_1$ and $\overline{w}_2$ which is a witness for $\overline{z} \in \overline{L}_2$. We obtain $\overline{\pi}$ from the challenger and use it to generate wzk_2. At the end of the interaction with prover P^*, we check if event E happens. If it happens, we output 1 and otherwise, we output 0.

We note that if $\overline{\pi}$ was generated using the witness $\overline{w}_1$, then the view of P^* in the above interaction is identically distributed to Hyb_4. Otherwise, it is identically distributed to Hyb_5. Thus, the probability that the above reduction outputs 1 in Hyb_4 is p_4 and the probability that it outputs 1 in Hyb_5 is at most $p_4 - \varepsilon(\lambda)$. Thus, the above reduction breaks the witness indistinguishability property of $(\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}})$ with advantage $\varepsilon(\lambda)$ (which is non-negligible) and this is a contradiction.

Lemma 7. *Assuming the soundness of* $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$*, we have* $p_9 \le \mu(\lambda)/2$*.*

Proof. Assume for the sake of contradiction that $p_9 \ge \mu(\lambda)/2$. We show that this contradicts the soundness of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$.

Note that if E happens in Hyb_9 then the following conditions hold:

1. $\mathsf{ZAP.Verify}(s, z, \pi) = 1$.
2. $w = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}')$ and $(\mathsf{stmt}, w) \notin R_L$.
3. $(s_1, s_2) = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_2)$ and $\overline{\mathsf{ct}} \neq \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \bot; s_1); s_2)$.
4. $f(\mathsf{Den.Dec}(\mathsf{sk}'.\mathsf{ct}'_1)) \neq y'$.
5. $\mathsf{com} = \mathsf{Com}(1^\lambda, 1)$

It now follows from the perfect correctness of $\mathsf{Den.Dec}$ and perfect binding of Com that $z \notin L_1 \vee L_2 \vee L_3 \vee L_4$. Hence, if $\mathsf{ZAP.Verify}(r, z, \pi) = 1$, then the above prover breaks the soundness of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ with non-negligible probability $\mu(\lambda)/2$ and this is a contradiction.

Weak Zero-Knowledge Let V^* be a malicious verifer, let D be a distinguisher, and let $\mathsf{InstGen}$ be an instance generator. Let ε be the distinguishing parameter.

Description of Simulator. We give the formal description of $\mathsf{Sim}_{\mathsf{wzk}}$.

- $\mathsf{Sim}_{\mathsf{wzk}}$ sets wzk_1 to be $(\mathsf{pk}', r, \overline{s}, y)$ which are sampled identically to $\mathsf{wZK.P}_1(1^\lambda)$.
- $\mathsf{Sim}_{\mathsf{wzk}}$ obtains wzk_2 and parses it as $(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{fpk}_1, \mathsf{pk}_2, \mathsf{ct}_2, \mathsf{fpk}_2, y', \mathsf{com}, \mathsf{fct}_1, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, s, \overline{\pi})$.
- **Step-1:** $\mathsf{Sim}_{\mathsf{wzk}}$ does the following:
  1. It constructs a homomorphic simulation circuit $\mathsf{HS}_2[s_1]$ which is described in Figure 4.
  2. $\mathsf{Sim}_{\mathsf{wzk}}$ parses fct_2 as $(\mathsf{fct}_{2,1}, \dots, \mathsf{fct}_{2,\lambda})$.
  3. For each $k \in [\lambda]$, it computes $\overline{\mathsf{fct}}_{2,k} = \mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s_{1,k}), \mathsf{fct}_{2,k}))$.
  4. It then computes $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{fpk}_2, \mathsf{HS}_2[s_1], (\overline{\mathsf{fct}}_{2,1}, \dots, \overline{\mathsf{fct}}_{2,\lambda}))) = \mathsf{fct}'$.
  5. It finally computes $\widetilde{\mathbf{CC}}_2(\mathsf{fct}') = x$. If $f(x) = y$, then it aborts.
- **Step-2:**
  1. It constructs a homomorphic simulation circuit HS_1 described in Figure 5.
  2. It computes $\mathsf{FHE.Eval}(\mathsf{fpk}_1, \mathsf{HS}_1, \mathsf{fct}_1) = \mathsf{fct}'$. It runs $\widetilde{\mathbf{CC}}_1(\mathsf{fct}') = \rho$. It then checks if $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$. If yes, it does the following:
     (a) It generates π as $\mathsf{ZAP.Prove}(s, z, \rho)$.
     (b) It generates ct' as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$.
     (c) It generates the rest of the messages as in the protocol and sends wzk_3.
- **Step-3:**
  1. If $f(x) \neq y$ and if $\mathsf{com} \neq \mathsf{Com}(1^\lambda, 0; \rho)$ then it does the following:
     (a) It generates $\overline{\mathsf{ct}}$ as $\mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \bot; s_1); s_2)$ where $s_1, s_2 \leftarrow \{0,1\}^\lambda$.

(b) It generates ct_2' as $\mathsf{Den.Enc}(\mathsf{pk}', (s_1, s_2); s_3)$ where $s_3 \leftarrow \{0,1\}^\lambda$.
(c) It computes $\pi \leftarrow \mathsf{ZAP.Prove}(r, z, (s_1, s_2, s_3))$.
(d) It generates $\mathsf{ct}' \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$.
(e) It generates the rest of the components in wzk_3 as before and sends it.

Proof of Indistinguishability. In the following, let p_i be the probability that D outputs 1 when it is given the distribution generated in Hyb_i as input.

– $\underline{\mathsf{REAL}}$: This corresponds to the view of the malicious verifier V^* during its interaction with the honest prover. That is, the output of Hyb_0 is identically distributed to $\mathsf{REAL}(1^\lambda, V^*, D, q, \mathsf{InstGen})$.
– $\underline{\mathsf{Hyb}_0}$: We receive $\mathsf{wzk}_2 = (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{fpk}_1, \mathsf{pk}_2, \mathsf{ct}_2, \mathsf{fpk}_2, y', \mathsf{com}, \mathsf{fct}_1, \mathsf{fct}_2,$ $\widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, s, \overline{\pi})$ from V^*. We construct a homomorphic simulation circuit $\mathsf{HS}_2[s_1](\cdot)$ described in Figure 4.
We parse fct_2 as $(\mathsf{fct}_{2,1}, \ldots, \mathsf{fct}_{2,\lambda})$.
  - For each $k \in [\lambda]$, we compute $\overline{\mathsf{fct}}_{2,k} = \mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s_{1,k}), \mathsf{fct}_{2,k}))$.
  - We compute $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{fpk}_2, \mathsf{HS}_2[s_1], (\overline{\mathsf{fct}}_{2,1}, \ldots, \overline{\mathsf{fct}}_{2,\lambda}))) = \mathsf{fct}'$.
  - We compute $\widetilde{\mathbf{CC}}_2(\mathsf{fct}') = x$.

  If $f(x) = y$, then we abort. In Lemma 8, we show that $|p_0 - p_{\mathsf{REAL}}| \leq \mathsf{negl}(\lambda)$ using the one-wayness of f.
– $\underline{\mathsf{Hyb}_2}$: We receive $\mathsf{wzk}_2 = (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{fpk}_1, \mathsf{pk}_2, \mathsf{ct}_2, \mathsf{fpk}_2, y', \mathsf{com}, \mathsf{fct}_1, \mathsf{fct}_2,$ $\widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, s, \overline{\pi})$ from V^*. We construct a homomorphic simulation circuit $\mathsf{HS}_1(\cdot)$ described in Figure 5. We compute $\mathsf{FHE.Eval}(\mathsf{fpk}_1, \mathsf{HS}_1, \mathsf{fct}_1) = \mathsf{fct}'$. We run $\widetilde{\mathbf{CC}}_1(\mathsf{fct}') = \rho$. We check if $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$. If yes, we make the following changes:
  1. $\underline{\mathsf{Hyb}_{1,1}}$: We generate π as $\mathsf{ZAP.Prove}(s, z, \rho)$. In Lemma 9, we show that using the witness indistinguishability of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ that $|p_1 - p_{1,1}| \leq \mathsf{negl}(\lambda)$.
  2. $\underline{\mathsf{Hyb}_{1,2}}$: We generate ct' as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$. In Lemma 10, we showing that using the security of $\mathsf{Den.Enc}$ that $|p_{1,2} - p_{1,1}| \leq \mathsf{negl}(\lambda)$.
– $\underline{\mathsf{Hyb}_3}$: If $f(x) \neq y$ and if $\mathsf{com} \neq \mathsf{Com}(1^\lambda, 0; \rho)$ then we make the following changes:
  1. $\underline{\mathsf{Hyb}_{2,1}}$: We switch $\overline{\mathsf{ct}}$ generated as part of wzk_3 to $\mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \perp; s_1); s_2)$ where $s_1, s_2 \leftarrow \{0,1\}^\lambda$. In Lemma 11, we show that $|p_{2,1} - p_2| \leq 4\varepsilon/5 + \mathsf{negl}(\lambda)$
  2. $\underline{\mathsf{Hyb}_{2,2}}$: We generate ct_2' in $\mathsf{wzk}_{3,i}$ as $\mathsf{Den.Enc}(\mathsf{pk}', (s_1, s_2); s_3)$ where $s_3 \leftarrow \{0,1\}^\lambda$. Via an identical argument to Lemma 10, we can use the security of $\mathsf{Den.Enc}$ to show that $|p_{2,2} - p_{2,1}| \leq \mathsf{negl}(\lambda)$.
  3. $\underline{\mathsf{Hyb}_{2,3}}$: We generate $\pi \leftarrow \mathsf{ZAP.Prove}(s, z, (s_1, s_2, s_3))$ (where π is part of $\mathsf{wzk}_{3,i}$). Via an identical argument to Lemma 9, we can rely on the witness indistinguishability of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ to prove that $|p_{2,2} - p_{2,3}| \leq \mathsf{negl}(\lambda)$.
  4. $\underline{\mathsf{Hyb}_{2,4}}$: We generate $\mathsf{ct}' \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$ where $\mathbf{0}$ is a default input. Again via an identical argument to Lemma 10, we can rely on the security of $\mathsf{Den.Enc}$ to show that $|p_{2,3} - p_{2,4}| \leq \mathsf{negl}(\lambda)$.

  This proves that $|p_{\mathsf{REAL}} - p_3| \leq 4\varepsilon/5 + \mathsf{negl}(\lambda)$. We note that Hyb_4 is identically distributed to the output of IDEAL using simulator Sim. Thus,

$$|\Pr[\mathsf{REAL}(1^\lambda, V^*, D, \mathsf{InstGen}) = 1] - \Pr[\mathsf{IDEAL}(1^\lambda, V^*, D, \mathsf{Sim}_{\mathsf{wzk}}, 1^{1/\varepsilon}, \mathsf{InstGen}) = 1]| \leq \varepsilon$$

Lemma 8. *Assuming the one-wayness of f, we have $|p_{\mathsf{REAL}} - p_0| \leq \mathsf{negl}(\lambda)$.*

Proof. Assume for the sake of contradiction that there exists a non-negligible function $\tau(\cdot)$ such that $|p_{\mathsf{REAL}} - p_0| \geq \tau(\lambda)$. We show that this contradicts the one-wayness of f.

Note that the only difference between Hyb_0 and Hyb_{11} is that if the conditions described in Hyb_1 happens then we abort. Thus, if $|p_{\mathsf{REAL}} - p_0| \geq \tau(\lambda)$, then the conditions described in Hyb_1 happens with probability at least $\tau(\lambda)$. We now argue that this breaks the one-wayness of f.

- **Hardcoded:** s_1 and the first and second round messages $(\mathsf{wzk}_1, \mathsf{wzk}_2)$.
- **Input:** ct'_1.
  1. It recovers the message x' from ct'_1 using pk' as the public key and s_1 as the randomness.
  2. It constructs a distinguisher D_2 that takes pk_2 and $\overline{\mathsf{ct}}$ as input where $\overline{\mathsf{ct}}$ is either an encryption under the public key pk_2 of $\mathsf{RSR.Enc}(\mathsf{pk}_1, \mathbf{0})$ or $\mathsf{RSR.Enc}(\mathsf{pk}_1, \perp)$. The distinguisher D_2 generates π using x', s_1 as the witness. It generates ct' as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$. It generates the rest of the messages in wzk_3 as in the protocol. D_2 provides wzk_3 to V^*. It finally runs the distinguisher D on the view of V^* and outputs whatever it outputs.
  3. It then runs $\mathsf{RSR.}\widetilde{\mathsf{Dec}}^{D_2}(\mathsf{pk}_2, \mathsf{ct}_2, 1^{5/\varepsilon})$ to obtain u_2 and outputs u_2.

Figure 4: Description of HS_2

- **Hardcoded:** Transcript in the first two rounds of the protocol.
- **Input:** ρ.
  1. It constructs a distinguisher D_1 takes as input $(\mathsf{pk}_1, \mathsf{ct}'')$ and ct'' is an RSR encryption under pk_1 of either $\mathbf{0}$ or $\perp$. It generates $\overline{\mathsf{ct}} = \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{ct}'')$. It generates π using the witness ρ. It generates ct' as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$. It generates the rest of the messages in wzk_3 as described in the protocol and gives this to V^*. It runs D on the view of V^* and outputs whatever D outputs.
  2. It then computes $u_1 = \mathsf{RSR.}\widetilde{\mathsf{Dec}}^{D_1}(\mathsf{pk}_1, \mathsf{ct}_1, 1^{5/\varepsilon})$ and outputs it.

Figure 5: Description of HS_1

We receive the one-wayness challenge y and use it to generate wzk_1. We then interact with V^* exactly like in Hyb_0. Once we receive wzk_2. We construct HS_2 as described in Hyb_1. We compute $\overline{\mathsf{fct}}_{2,k} = \mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s_{1,k}), \mathsf{fct}_{2,k}))$ for each $k \in [\lambda]$ (where s_1 is uniformly chosen). We then run $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{fpk}_2, \mathsf{HS}_2[s_1], (\overline{\mathsf{fct}}_{2,1}, \ldots, \overline{\mathsf{fct}}_{2,\lambda}))$ to obtain fct'. We compute $\widetilde{\mathbf{CC}}_2(\mathsf{fct}')$ to obtain x. If $f(x) = y$, then we output x as the pre-image of y.

Note that the probability that we find a valid pre-image is at least $\tau(\lambda)$ and this is a non-negligible function. This contradicts the one-wayness of f.

Lemma 9. *Assuming the witness indistinguishability of* $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$*, we have* $|p_1 - p_{1,1}| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists a non-negligible function $\tau(\cdot)$ such that $|p_1 - p_{1,1}| \geq \tau(\lambda)$. We now argue that this contradicts the witness indistinguishability of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$.

Note that the only difference between Hyb_1 and $\mathsf{Hyb}_{1,1}$ is that in Hyb_1, π is computed using the witness (w, s_4) whereas in $\mathsf{Hyb}_{1,1}$, it is computed using the witness ρ. We interact with the ZAP challenger and provide s, the instance z and $w_0 = (w, s_4)$ and $w_1 = \rho$ as the two challenge witnesses. We obtain π from the challenger and use this to generate wzk_3.

We observe that if π was generated using the witness w_0 then the view of V^* in the above interaction is identical to Hyb_1. Otherwise, it is identically distributed to $\mathsf{Hyb}_{1,1}$. Thus, if $|p_1 - p_{1,1}| \geq \tau(\lambda)$, we break the witness indistinguishability of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ with non-negligible advantage $\tau(\lambda)$ and this is a contradiction.

Lemma 10. *Assuming the security of* $\mathsf{Den.Enc}$*, we have* $|p_{1,2} - p_{1,1}| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists a non-negligible function $\tau(\cdot)$ such that $|p_{1,2} - p_{1,1}| \geq \tau(\lambda)$. We now argue that this contradicts the security of $\mathsf{Den.Enc}$.

Note that the only difference between $\mathsf{Hyb}_{1,2}$ and $\mathsf{Hyb}_{1,1}$ is that in $\mathsf{Hyb}_{1,2}$, ct' is generated as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$ whereas in $\mathsf{Hyb}_{1,1}$, it is generated as $\mathsf{Den.Enc}(\mathsf{pk}', w)$. We interact with the security challenger and provide $w, \mathbf{0}$ as the two challenge message. We obtain pk' and ct' from the challenger use this to generate the view of V^*.

We observe that if ct' was generated as an encryption of w then the view of V^* in the above interaction is identical to $\mathsf{Hyb}_{1,1}$. Otherwise, it is identically distributed to $\mathsf{Hyb}_{1,2}$. Thus, if $|p_{1,2} - p_{1,1}| \geq \tau(\lambda)$, we break the security of $\mathsf{Den.Enc}$ with non-negligible advantage $\tau(\lambda)$ and this is a contradiction.

Lemma 11. *Assuming the witness indistinguishability of* $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ *and security of* $\mathsf{Den.Enc}$*, we have* $|p_{2,1} - p_2| \leq 4\varepsilon/5 + \mathsf{negl}(\lambda)$.

Proof. In order to prove this, we show that if

$$|\Pr[D \text{ o/p } 1 \text{ when } \overline{\mathsf{ct}} \leftarrow \mathsf{RSR.Enc}(pk_2, \mathsf{RSR.Enc}(pk_1, \mathbf{0}))] - \Pr[D \text{ o/p } 1 \text{ when } \overline{\mathsf{ct}} \leftarrow \mathsf{RSR.Enc}(pk_2, \mathsf{RSR.Enc}(pk_1, \bot))]| \geq 3\varepsilon/5$$

where $\overline{\mathsf{ct}}$ is the ciphertext generated as part of wzk_3 (which is in turn generated as in the honest prover description) then either $f(x) = y$ or $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$ except with probability $\varepsilon/5 + \mathsf{negl}(\lambda)$. Thus, with probability at least $1 - (\varepsilon/5 + \mathsf{negl}(\lambda))$ if $f(x) \neq y$ and $\mathsf{com} \neq \mathsf{Com}(1^\lambda, 0; \rho)$, we have that the above equation does not hold. This shows that $|p_{2,1} - p_2| \leq 4\varepsilon/5 + \mathsf{negl}(\lambda)$.

To prove the above statement, we first observe from the soundness of ZAP system $(\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}})$ that $\overline{z} \in \overline{L_1}$ or $\overline{z} \in \overline{L}_2$ except with negligible probability (which we term as the bad event). Conditioning on this bad event not happening, we consider the two cases:

- **Case-1:** $\overline{z} \in \overline{L}_1$**:** In this case, we show that ρ that is extracted in Hyb_2 is such that $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$ except with negligible probability. To see this, we consider a sequence of hybrids:
  - Hyb'_1 : In this hybrid, we consider a modified homomorphic simulation circuit HS'_1 that is same as HS_1 except that the corresponding distinguisher D'_1 generates ct', π in wzk_3 using the witness (w, s_4). We run $\mathsf{FHE.Eval}(\mathsf{fpk}_1, \mathsf{HS}'_1, \mathsf{fct}_1)$ to obtain fct'. We then run $\widetilde{\mathbf{CC}}(\mathsf{fct}')$ to compute ρ and if $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$ then we output win. We now argue that the probability of not outputting win in Hyb'_1 is negligible. By assumption, D is able to distinguish $\overline{\mathsf{ct}} \leftarrow \mathsf{RSR.Enc}(pk_2, \mathsf{RSR.Enc}(pk_1, \mathbf{0}))$ and $\overline{\mathsf{ct}} \leftarrow \mathsf{RSR.Enc}(pk_2, \mathsf{RSR.Enc}(pk_1, \bot))$ with advantage more than $\varepsilon/5$. This means that D'_1 also distinguishes between $\mathsf{RSR.Enc}(\mathsf{pk}_1, \mathbf{0})$ and $\mathsf{RSR.Enc}(\mathsf{pk}_1, \bot)$ with the same advantage. Thus, from the property of $\mathsf{RSR.}\widetilde{\mathsf{Dec}}$ that the output of HS_1 on input ρ is u_1 with overwhelming probability. Since $\overline{z} \in \overline{L}_1$, it follows from the perfect correctness of $\mathcal{O}$ that the output of $\widetilde{\mathbf{CC}}(\mathsf{fct}')$ is ρ such that $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$.
  - Hyb'_2 : In this hybrid, we let D'_1 to use ρ as the witness instead (w, s_4). Via an identical argument to Lemma 9, we can show that $\mathsf{Hyb}'_2 \approx_c \mathsf{Hyb}'_3$.
  - Hyb'_3 : In this hybrid, we let D'_1 to generate ct' as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0}; s_4)$ (where s_4 is uniformly chosen). Via an identical argument to Lemma 10, we can show that $\mathsf{Hyb}'_2 \approx_c \mathsf{Hyb}'_3$. Note that Hyb'_3 is identically distributed to HS_1 used by simulator.
- **Case-2:** $\overline{z} \in \overline{L}_2$**:** In this case, we show that $f(x) = y$ except with probability $\varepsilon/5 + \mathsf{negl}(\lambda)$. To see this, we consider a sequence of hybrids:
  - Hyb'_1 : We consider a modified homomorphic simulation circuit $\mathsf{HS}'_2[\mathbf{0}]$. HS'_2 is same as HS_2 except that the corresponding distinguisher D'_2 uses the witness w, s_4 to generate ct' as well as the proof π in wzk_3. We consider the following computation
    * For each $k \in [\lambda]$, compute $\overline{\mathsf{fct}}_{2,k} = \mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s_{1,k}), \mathsf{FHE.Enc}(\mathsf{fpk}_2, 0)))$.
    * Compute $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{fpk}_2, \mathsf{HS}'_2[\mathbf{0}], (\overline{\mathsf{fct}}_{2,1}, \ldots, \overline{\mathsf{fct}}_{2,\lambda}))) = \mathsf{fct}'$.
    * Compute $\widetilde{\mathbf{CC}}_2(\mathsf{fct}') = x$.

    If $f(x) = y$, we output the special symbol win. We show that with overwhelming probability that we output win in Hyb'_1. By assumption, D is able to distinguish $\overline{\mathsf{ct}} \leftarrow \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \mathbf{0}))$ and $\overline{\mathsf{ct}} \leftarrow \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \bot))$ with advantage more than $\varepsilon/5$. This means that D'_2 (in the above construction) also distinguishes between these two ciphertexts with the same advantage. Thus, from the property of $\widetilde{\mathsf{RSR.Dec}}$ that the output of HS_2' on input ciphertexts encrypting 0 is u_2 with overwhelming probability. Since $\overline{z} \in \overline{L}_2$, it follows from the perfect correctness of $\mathcal{O}$ that the output of $\widetilde{\mathbf{CC}}(\mathsf{fct}')$ is x such that $f(x) = y$.

- $\mathsf{Hyb}'_{1,j}$: For each $j \in [\lambda+1]$, in $\mathsf{Hyb}'_{1,j}$ we make the following changes:
  - ∗ For each $k \geq j$, we compute $\overline{\mathsf{fct}}_{2,k} = \mathsf{Sanitize}(\mathsf{FHE.Eval}(\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s_{1,k}), \mathsf{FHE.Enc}(\mathsf{fpk}_2, 0)))$. For each $k < j$, we compute $\overline{\mathsf{fct}}_{2,k} = \mathsf{Sanitize}(\mathsf{FHE.Eval}(\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s_{1,k}), \mathsf{fct}_{2,k})$.

  In Claim 4.4, we show that $\mathsf{Hyb}'_{1,j} \approx_{\varepsilon/5\lambda + \mathsf{negl}(\lambda)} \mathsf{Hyb}'_{1,j+1}$.
- Hyb'_2 : In this hybrid, we compute fct' as $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{fpk}_2, \mathsf{HS}'_2[s_1], (\overline{\mathsf{fct}}_{2,1}, \ldots, \overline{\mathsf{fct}}_{2,\lambda})))$. Since the output of $\mathsf{HS}'_2[s_1]$ is identical to the output of $\mathsf{HS}'_2[\mathbf{0}]$, it follows from the statistical circuit privacy that $\mathsf{Hyb}'_{1,\lambda+1} \approx_s \mathsf{Hyb}'_2$.
- Hyb'_3 : In this hybrid, we modify HS'_2 so that the corresponding distinguisher uses $x', s_1 = (s_{1,1}, \ldots, s_{1,\lambda})$ as the witness to generate π in wzk_3. From the witness indistinguishability of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$, we have $\mathsf{Hyb}'_2 \approx_c \mathsf{Hyb}'_3$.
- Hyb'_4 : In this hybrid, we modify HS'_2 to generate ct' as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0}; s_4)$ (where s_4 is uniformly chosen). Via an identical argument to Lemma 10, we can show that $\mathsf{Hyb}'_3 \approx_c \mathsf{Hyb}'_4$. Observe that Hyb'_4 is identical to the setting where we use $\mathsf{HS}_2[s_1]$ instead.

Claim. Assuming the security of $\mathsf{Den.Enc}$, we have $\mathsf{Hyb}'_{1,j} \approx_{\varepsilon/5\lambda+\mathsf{negl}(\lambda)} \mathsf{Hyb}'_{1,j+1}$ for each $j \in [0, \lambda]$.

Proof. We show this via a sequence of sub-hybrids.

– $\mathsf{Hyb}'_{1,j,1}$: In this hybrid, we generate $\overline{\mathsf{fct}}_j$ as $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Enc}(\mathsf{fpk}_2, \mathsf{Den.Enc}(\mathsf{pk}', 0; s_{1,j}))))$. It follows from the statistical circuit privacy property of $\mathsf{Sanitize}$ that the probability we output win in $\mathsf{Hyb}'_{1,j}$ is statistically close to its probability in $\mathsf{Hyb}'_{1,j,1}$.

– $\mathsf{Hyb}'_{1,j,2}$: In this hybrid, once we receive $\mathsf{wzk}_{2,i}$ from V^*, we inefficiently compute two bits: the first bit indicates whether $\overline{z} \in \overline{L}_2$ and the second bit gives x'_j. We now use the leakage lemma (see Theorem 3) to construct a simulator h of size $\mathsf{poly}(\lambda)2^2 \cdot (\varepsilon/10\lambda)^{-2}$ to simulate this leakage such that the difference between the probability that we output win in $\mathsf{Hyb}'_{1,j,2}$ and $\mathsf{Hyb}'_{1,j,1}$ is $\varepsilon/10\lambda$.

– $\mathsf{Hyb}'_{1,j,3}$: In this hybrid, if $\overline{z} \in \overline{L}_2$, we compute $\overline{\mathsf{fct}}_j$ as $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Enc}(\mathsf{fpk}_2, \mathsf{Den.Enc}(\mathsf{pk}', x'_j; s_{1,j}))))$. The difference between the probabilities that we output win in this hybrid and the previous hybrid is at most $\mathsf{negl}(\lambda)$ and this follows from the security of $\mathsf{Den.Enc}$.

– $\mathsf{Hyb}'_{1,j,4}$ and $\mathsf{Hyb}'_{1,j,5}$: In these two hybrids, we reverse the changes made in $\mathsf{Hyb}'_{1,j,2}$ and $\mathsf{Hyb}'_{1,j,1}$. We note that $\mathsf{Hyb}'_{1,j,5}$ is identically distributed to $\mathsf{Hyb}'_{1,j+1}$.

5 Three-Round ε-secure Protocol for Inputless Functionalities

In this section, we give our construction of a three-round 2PC protocol that achieves ε-security for computing inputless functionalities. In Section 5.1, we give the formal definition. In Section 5.2 we describe the building blocks and in Section 5.3 we describe our construction.

5.1 Definition

We start with the definition of a three-round secure two-party computation protocol for computing an inputless functionality g that has standard security against malicious senders and ε-security against malicious receivers.

Definition 9. *A three-round protocol $\Pi = (\Pi_1, \Pi_2, \Pi_3)$ between a sender and a receiver is said to compute an inputless functionality g with security against malicious receivers and ε-security against malicious senders, if:*

- ***Security against Malicious Senders.*** *We require the existence of an (expected) PPT simulator* Sim_S *such that for any adversary* $\mathcal{A}$ *corrupting the sender, the view of the adversary* $\mathcal{A}$ *and the output of the honest receiver in the real execution of the protocol is computationally indistinguishable to the ideal experiment with the simulator* Sim_S *that has oracle access to the functionality* g *and the adversary* $\mathcal{A}$ *and can instruct this functionality to either deliver the output to the receiver or* $\perp$*.*
- ε***-Security against Malicious Receivers.*** *For any (stateful) adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *corrupting the receiver, there exists a (stateful) simulator* Sim_R *such that for any non-negligible error parameter* ε*, we have:*

$$|\Pr[\mathsf{REAL}(1^\lambda, \mathcal{A}, 1^{1/\varepsilon}) = 1] - \Pr[\mathsf{IDEAL}(1^\lambda, \mathcal{A}, \mathsf{Sim}_R, 1^{1/\varepsilon}) = 1] \le \varepsilon$$

where REAL *and* IDEAL *experiments are described in Figure 6.*

$\mathsf{REAL}(1^\lambda, \mathcal{A}, 1^{1/\varepsilon})$	$\mathsf{IDEAL}(1^\lambda, \mathcal{A}, \mathsf{Sim}_R, 1^{1/\varepsilon})$
– $\pi_1 \leftarrow \Pi_1(1^\lambda)$.	– $\pi_1 \leftarrow \mathsf{Sim}_R(1^\lambda, 1^{1/\varepsilon})$.
– $\pi_2 \leftarrow \mathcal{A}_1(\pi_1)$.	– $\pi_2 \leftarrow \mathcal{A}(\pi_1)$.
– $\pi_3 \leftarrow \Pi_3(\pi_1, \pi_2)$	– $\pi_3 \leftarrow \mathsf{Sim}_R^g(\pi_1, \pi_2, 1^{1/\varepsilon})$
– Output $\mathcal{A}_2(\pi_1, \pi_2, \pi_3)$.	– Output $\mathcal{A}_2(\pi_1, \pi_2, \pi_3)$.

Figure 6: Descriptions of REAL and IDEAL for 2PC.

5.2 Building Blocks

Let g be the inputless functionality. For simplicity, we denote g as a two-party functionality that takes randomness x_1 and x_2 and uses $x_1 \oplus x_2$ to sample from the underlying distribution. The construction makes use of the following building blocks.

- A dense public-key encryption scheme ($\mathsf{Den.Gen}, \mathsf{Den.Enc}, \mathsf{Den.Dec}$). We assume without loss of generality that the encryption is done bit-by-bit.
- A symmetric key Encryption ($\mathsf{SKEnc}, \mathsf{SKDec}$).
- A fully-homomorphic encryption scheme ($\mathsf{Setup}, \mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}$, $\mathsf{FHE.Eval}, \mathsf{Sanitize}$) that is statistically circuit-private. We assume without loss of generality that the encryption is done bit-by-bit.
- A one-way permutation $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$.
- A three-round trapdoor generation protocol ($\mathsf{TD}_1, \mathsf{TD}_2, \mathsf{TD}_3, \mathsf{TDVerify}$).
- A non-interactive commitment Com.
- A three-round extractable commitment ($\mathsf{ExtCom}_1, \mathsf{ExtCom}_2, \mathsf{ExtCom}_3$)
- A compute and compare obfuscation $\mathcal{O}$.
- A random self-reducible public-key encryption ($\mathsf{RSR.Gen}, \mathsf{RSR.Enc}, \mathsf{RSR.Dec}, \mathsf{RSR}.\widetilde{\mathsf{Dec}}$).
- A secure function evaluation ($\mathsf{SFE}_1, \mathsf{SFE}_2, \mathsf{out}$) for computing the function g.
- A ZAP proof ($\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}}$) for the NP language $\overline{L} = \overline{L}_1 \vee \overline{L}_2$ where $\overline{L}_1$ and $\overline{L}_2$ consists of instances

$$\overline{z} = (y, y', r, \widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, \mathsf{sfe}_1, \mathsf{ct}_1, \mathsf{fpk}_1, \mathsf{pk}_1, \mathsf{ct}_2, \mathsf{pk}_2, \mathsf{fpk}_2, \mathsf{fct}_1, \mathsf{fct}_2, \mathsf{com}) \tag{5.1}$$

such that:

- $\overline{z} \in \overline{L}_1$ iff

$$\exists (x_2, u_1, \rho, r_1, r_2, r_3, r_4, r_5, r_6) \quad \text{s.t.} \quad \begin{array}{l} \mathsf{sfe}_1 = \mathsf{SFE}(x_2; r_1) \wedge \\ (\mathsf{pk}_1, \mathsf{sk}_1) = \mathsf{RSR.Gen}(1^\lambda; r_2) \wedge \\ \mathsf{ct}_1 = \mathsf{RSR.Enc}(\mathsf{pk}_1, u_1; r_3) \wedge \\ (\mathsf{fpk}_1, \mathsf{fsk}_1) = \mathsf{FHE.Gen}(r; r_4) \wedge \\ \mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho) \wedge \\ \mathsf{fct}_1 = \mathsf{FHE.Enc}(\mathsf{fpk}_1, \rho; r_5) \wedge \\ \widetilde{\mathbf{CC}}_1 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_1, \cdot), u_1, (m, r_1, \rho)]; r_6) \end{array}$$

- $\overline{z} \in \overline{L_2}$ iff

$$\exists (x, u_2, x', r_1, r_2, r_3, r_4, r_5) \quad \text{s.t.} \quad \begin{array}{l} \mathsf{TDVerify}(\mathsf{td}_1, x) = 1 \wedge \\ (\mathsf{pk}_2, \mathsf{sk}_2) = \mathsf{RSR.Gen}(1^\lambda; r_1) \wedge \\ \mathsf{ct}_2 = \mathsf{RSR.Enc}(\mathsf{pk}_2, u_2; r_2) \wedge \\ (\mathsf{fpk}_2, \mathsf{fsk}_2) = \mathsf{FHE.Gen}(r; r_3) \wedge \\ y' = f(x') \wedge \\ \mathsf{fct}_2 = \mathsf{FHE.Enc}(\mathsf{fpk}_2, x'; r_4) \wedge \\ \widetilde{\mathbf{CC}}_2 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x]; r_5) \end{array}$$

– A ZAP proof ($\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify}$) for the NP language $L = L_1 \vee L_2 \vee L_3 \vee L_4$ where L_1, L_2, L_3, and L_4 consists of instances

$$z = (\mathsf{pk}', \mathsf{vk}, y', \mathsf{ct}', \mathsf{ct}'_1, \mathsf{ct}'_2, \overline{\mathsf{ct}}, \mathsf{sfe}_1) \tag{5.2}$$

such that:

- $z \in L_1$ iff

$$\exists (x', s'_1) \quad \text{s.t.} \quad \begin{array}{l} y' = f(x') \wedge \\ \mathsf{ct}'_1 = \mathsf{Den.Enc}(\mathsf{pk}', x'; s'_1) \wedge \end{array}$$

- $z \in L_2$ iff

$$\exists (s'_1, s'_2, s'_3) \quad \text{s.t.} \quad \begin{array}{l} \mathsf{ct}'_2 = \mathsf{Den.Enc}(\mathsf{pk}', (s'_1, s'_2); s'_3) \wedge \\ \overline{\mathsf{ct}} = \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \perp; s'_1); s'_2) \end{array}$$

- $z \in L_3$ iff

$$\exists (x_1, s_2, s_3) \quad \text{s.t.} \quad \begin{array}{l} \mathsf{sfe}_2 = \mathsf{SFE}_2(\mathsf{sfe}_1, g(x_1, \cdot); s_3) \wedge \\ \mathsf{ct}' = \mathsf{Den.Enc}(\mathsf{pk}', (\mathsf{sfe}_2, x_1); s_2) \wedge \end{array}$$

- $z \in L_4$ iff

$$\exists \rho \quad \text{s.t.} \quad \mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$$

– A three-round delayed input weak zero-knowledge protocol ($\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2$) for the NP language L' where L' consists of instances of the form

$$z' = (\mathsf{pk}', \mathsf{ct}', y, \overline{\mathsf{ct}}, \mathsf{sfe}_1, \mathsf{ECom}_1, \widehat{\mathsf{ct}}) \tag{5.3}$$

such that $z' \in L'$ iff

$$\exists(\mathsf{sfe}_2, \mathsf{ssk}, x_1, \widehat{s}, \{s_i\}_{i\in[6]}) \qquad \text{s.t.} \qquad \begin{aligned} &(\mathsf{pk}', \cdot) = \mathsf{Den.Enc}(1^\lambda; s_1) \wedge \\ &\mathsf{ECom}_1 = \mathsf{ExtCom}(1^\lambda, s_1; s_6) \wedge \\ &\mathsf{ct}' = \mathsf{Den.Enc}(\mathsf{pk}', (\mathsf{sfe}_2, x_1); s_2) \wedge \\ &\widehat{\mathsf{ct}} = \mathsf{SKEnc}(\mathsf{ssk}, \mathsf{sfe}_2; \widehat{s}) \wedge \\ &\mathsf{sfe}_2 = \mathsf{SFE}_2(\mathsf{sfe}_1, g(x_1, \cdot); s_3) \wedge \\ &\overline{\mathsf{ct}} = \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \mathsf{ssk}; s_4); s_5) \end{aligned}$$

5.3 Construction

We describe the first three rounds of our protocol in Figure 7 and we describe the output computation below.

Output Computation. To compute the output, the receiver does the following:

1. It checks if $\mathsf{ZAP.Verify}(s, z, \pi) = 1$ and $\mathsf{wZK.V}_2(z', \tau_2, (\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3)) = 1$. If not, it outputs $\perp$.
2. It checks if $(\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3)$ and $(\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3)$ are valid transcripts.
3. It computes $\mathsf{ssk} = \mathsf{RSR.Dec}(\mathsf{sk}_1, \mathsf{RSR.Dec}(\mathsf{sk}_2, \overline{\mathsf{ct}}))$, computes $\mathsf{sfe}_2 = \mathsf{SKDec}(\mathsf{ssk}, \widehat{\mathsf{ct}})$ and checks if $\mathsf{sfe}_2 = \perp$. If yes, it outputs $\perp$.
4. Else, it computes $\sigma = \mathsf{out}(\mathsf{sfe}_2, (x_2, r_1))$ and outputs σ.

5.4 Proof of Security

In this section, we show that the construction described in Figure 7 is a ε-secure protocol for computing the inputless functionality g.

5.5 Sender is Corrupt

Let $\mathcal{A}$ be an adversary that corrupts the sender. We give the description of the simulator Sim_R below.

Description of Sim_R.

1. Sim_R interacts with the adversary $\mathcal{A}$ as per the honest protocol description using an uniformly chosen x_2. If the adversary aborts or fails to send an incorrect message, then Sim_R instructs the ideal functionality g to output $\perp$ to the honest receiver and outputs the view of $\mathcal{A}$.
2. If Sim_R doesn't abort at the end of the interaction, then Sim_R estimates the probability that $\mathcal{A}$ generates a valid transcript. Specifically, it rewinds and sends independently sampled second round messages and waits until it obtains 12λ valid transcripts. Let m be the number of trials needed until the $\mathcal{A}$ produces 12λ accepting transcripts. It sets $\widetilde{\varepsilon} = 12\lambda/m$. While estimating this quantity, Sim_R in parallel runs the rewinding extractor on the extractable commitment as well the trapdoor extractor to obtain (s_1, x).
3. It repeats the following for $\lambda^2/\widetilde{\varepsilon}$ times:
   (a) It generates ct_1 as $\mathsf{RSR.Enc}(\mathsf{pk}_1, 0^\lambda)$.
   (b) It generates $\widetilde{\mathbf{CC}}_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|\mathsf{FHE.Dec}(\mathsf{fsk}_1, \cdot)|}, 1^\lambda)$.
   (c) It generates fct_1 as $\mathsf{FHE.Enc}(\mathsf{fpk}_1, 0^\lambda)$ and com as $\mathsf{Com}(1^\lambda, 1)$.
   (d) It computes $(\mathsf{pk}_2, \mathsf{sk}_2) = \mathsf{RSR.Gen}(1^\lambda; r_1)$ where $r_1 \leftarrow \{0,1\}^\lambda$.
   (e) It then computes $\mathsf{ct}_2 = \mathsf{RSR.Enc}(\mathsf{pk}_2, u_2; r_2)$ where $r_2, u_2 \leftarrow \{0,1\}^\lambda$.
   (f) It then samples $(\mathsf{fpk}_2, \mathsf{fsk}_2) = \mathsf{FHE.Gen}(1^\lambda; r_3)$
   (g) It computes $\mathsf{fct}_2 = \mathsf{FHE.Enc}(\mathsf{fpk}_2, x'; r_4)$
   (h) It computes $\widetilde{\mathbf{CC}}_2 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x]; r_5)$
   (i) It then generates $\overline{\pi} \leftarrow \overline{\mathsf{ZAP.Prove}}(\overline{r}, \overline{z}, (x, u_2, x', r_1, r_2, r_3, r_4, r_5))$.
   (j) It generates the rest of the second round messages as in the protocol and sends it to $\mathcal{A}$. It computes $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Den.Gen}(1^\lambda; s_1)$.

- **First Round:** The sender does the following:
  1. It samples $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Den.Gen}(1^\lambda; s_1)$ (where $s_1 \leftarrow \{0,1\}^\lambda$) and $r \leftarrow \mathsf{Setup}(1^\lambda)$.
  2. It computes $\mathsf{ECom}_1 \leftarrow \mathsf{ExtCom}(1^\lambda, s_1; s_6)$ where $s_6 \leftarrow \{0,1\}^\lambda$.
  3. It samples a uniform random string $\overline{s} \leftarrow \{0,1\}^*$.
  4. It samples $\mathsf{td}_1 \leftarrow \mathsf{TD}_1(1^\lambda)$.
  5. It samples $\mathsf{wzk}_1 \leftarrow \mathsf{wZK.P}_1(1^\lambda)$.
  6. It sends $(\mathsf{pk}', \mathsf{ECom}_1, \mathsf{td}_1, r, \overline{s}, y, \mathsf{wzk}_1)$.
- **Second Round:** The receiver does the following:
  1. It computes $\mathsf{sfe}_1 := \mathsf{SFE}(x_2; r_1)$ where $x_2, r_1 \leftarrow \{0,1\}^\lambda$.
  2. It samples $(\mathsf{pk}_1, \mathsf{sk}_1) := \mathsf{RSR.Gen}(1^\lambda; r_2)$ where $r_2 \leftarrow \{0,1\}^\lambda$.
  3. It samples $u_1 \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{ct}_1 := \mathsf{RSR.Enc}(\mathsf{pk}_1, u_1; r_3)$ where $r_3 \leftarrow \{0,1\}^\lambda$.
  4. It samples $(\mathsf{fpk}_1, \mathsf{fpk}_2) := \mathsf{FHE.Gen}(r; r_4)$ where $r_4 \leftarrow \{0,1\}^\lambda$.
  5. It computes $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$ where $\rho \leftarrow \{0,1\}^\lambda$.
  6. It then computes $\mathsf{fct}_1 = \mathsf{FHE.Enc}(\mathsf{fpk}_1, (x_2, r_1, \rho); r_5)$ where $r_5 \leftarrow \{0,1\}^\lambda$.
  7. It computes $\widetilde{\mathbf{CC}}_1 := \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_1, \cdot), u_1, (x_2, r_1, \rho)]; r_6)$ where $r_6 \leftarrow \{0,1\}^\lambda$.
  8. It samples $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{RSR.Enc}(1^\lambda)$ and computes $\mathsf{ct}_2 \leftarrow \mathsf{RSR.Enc}(\mathsf{pk}_2, 0^\lambda)$.
  9. It samples $x' \leftarrow \{0,1\}^\lambda$ and computes $y' = f(x')$.
  10. It samples $(\mathsf{fpk}_2, \mathsf{fsk}_2) \leftarrow \mathsf{FHE.Gen}(r)$ and generates $\mathsf{fct}_2 \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}_2, 0^\lambda)$.
  11. It computes $\widetilde{\mathbf{CC}}_2 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot)|}, 1^\lambda)$.
  12. It computes $\overline{\pi} \leftarrow \overline{\mathsf{ZAP.Prove}}(\overline{s}, \overline{z}, (x_2, u_1, \rho, \{r_i\}_{i\in[6]}))$ (where $\overline{z}$ is described in Equation 5.1).
  13. It samples a uniform random string $s \leftarrow \{0,1\}^*$ and computes $(\mathsf{wzk}_2, \tau_2) \leftarrow \mathsf{wZK.V}_1(1^\lambda, \mathsf{wzk}_1)$.
  14. It samples $\mathsf{ECom}_2 \leftarrow \mathsf{ExtCom}(\mathsf{ECom}_1)$ and $\mathsf{td}_2 \leftarrow \mathsf{TD}_2(\mathsf{td}_1)$.
  15. It sends $(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{fpk}_1, \mathsf{pk}_2, \mathsf{ct}_2, \mathsf{fpk}_2, y', \mathsf{sfe}_1, \mathsf{fct}_1, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, s,$ $\mathsf{wzk}_2, \overline{\pi}, \mathsf{ECom}_2, \mathsf{td}_2)$.
- **Third Round:** The sender does the following:
  1. It checks if $\overline{\mathsf{ZAP.Verify}}(\overline{s}, \overline{z}, \overline{\pi}) = 1$ and aborts otherwise.
  2. It computes $\mathsf{sfe}_2 := \mathsf{SFE}_2(\mathsf{sfe}_1, g(x_1, \cdot); s_3)$ where $x_1, s_3 \leftarrow \{0,1\}^\lambda$.
  3. It samples $\mathsf{ssk} \leftarrow \{0,1\}^\lambda$.
  4. It computes $\mathsf{ct}' \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', (\mathsf{sfe}_2, x_1); s_2)$ where $s_2 \leftarrow \{0,1\}^\lambda$.
  5. It computes $\overline{\mathsf{ct}} := \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \mathsf{ssk}; s_4); s_5)$ where $s_4, s_5 \leftarrow \{0,1\}^\lambda$.
  6. It generates $\widehat{\mathsf{ct}} := \mathsf{SKEnc}(\mathsf{ssk}, \mathsf{sfe}_2; \widehat{s})$ where $\widehat{s} \leftarrow \{0,1\}^\lambda$.
  7. It generates $\mathsf{ct}'_1 \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', 0^\lambda)$ and $\mathsf{ct}'_2 \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', 0^{2\lambda})$.
  8. It computes $\pi \leftarrow \mathsf{ZAP.Prove}(s, z, (\{x_1, s_2, s_3\}))$ (where z is described in Equation 5.2).
  9. It computes $\mathsf{wzk}_3 \leftarrow \mathsf{wZK.P}_2(\mathsf{wzk}_1, \mathsf{wzk}_2, (z', \mathsf{sfe}_2, x_1, \widehat{s}, \{s_i\}_{i\in[6]}))$ (where z' appears in Equation 5.3).
  10. It generates $\mathsf{ECom}_3 \leftarrow \mathsf{ExtCom}_3(\mathsf{ECom}_2, s_1; s_6)$ and $\mathsf{td}_3 \leftarrow \mathsf{TD}_3(\mathsf{td}_1, \mathsf{td}_2)$
  11. It sends $(\pi, \mathsf{wzk}_3, \mathsf{ct}', \mathsf{ct}'_1, \mathsf{ct}'_2, \overline{\mathsf{ct}}, \widehat{\mathsf{ct}}, \mathsf{ECom}_3, \mathsf{td}_3)$.

Figure 7: Three-Round Secure 2PC

(k) On receiving the last round message from $\mathcal{A}$, it checks if:
   i. $\mathsf{wZK.V}_2(z', \tau_2, (\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3)) = 1$.
   ii. It computes $(s_4, s_5) := \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_2)$ and checks if $\overline{\mathsf{ct}} \neq \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \bot; s_4); s_5)$.
   iii. It checks if $\mathsf{ZAP.Verify}(s, z, \pi) = 1$.

   If any of these checks fails, it repeats the iteration.

4. Finally, if it fails in each of these iterations or if in the accepting iteration we have $f(\mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_1)) = y'$, then it outputs a special symbol ABORT.
5. Otherwise, it asks the ideal functionality to deliver the output to the honest receiver and outputs the view of the corrupt sender.

The running time of the simulator can be shown to be expected polynomial time using standard techniques (see for instance [GK96]).

We show that the real and the ideal experiments are indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the view of $\mathcal{A}$ and the output of the honest receiver in the coin tossing protocol.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, in the output computation of the receiver, we make the following changes.
  1. We check if $\mathsf{wZK.V}_2(z', \tau_2, (\mathsf{wzk}_1, \mathsf{wzk}_2, \mathsf{wzk}_3)) = 1$.
  2. If it is the case, we check if $z' \in L'$ or not. If $z' \notin L'$, then we abort.

  In Lemma 12, we show from the adaptive computational soundness of $(\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2)$ that Hyb_0 and Hyb_1 are computationally indistinguishable.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:
  1. We non-uniformly fix the first round message of the receiver. We get as non-uniform advice s_1 which is the message inside ECom_1. We also non-uniformly compute x such that $\mathsf{TDVerify}(\mathsf{td}_1, x) = 1$.
  2. On receiving the final round message, we check if we $z' \in L'$. If it is the case, we set $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Den.Gen}(1^\lambda; s_1)$.
  3. We compute $(s_4, s_5) := \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_2)$.
  4. We check if $\overline{\mathsf{ct}} := \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \bot; s_4); s_5)$. If it is the case, we abort and output $\bot$.
  5. Instead of using $\mathsf{sk}_1, \mathsf{sk}_2, \widehat{\mathsf{ct}}$ to obtain sfe_2, we compute $(\mathsf{sfe}_2, x_1) := \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}')$.

  In Lemma 13, we show that Hyb_1 and Hyb_2 are identically distributed.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we reverse the changes made in Hyb_1. That is, we do not abort if $z' \notin L'$. Via and identical argument to Lemma 12, we can show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.
- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we instruct the receiver to output a special symbol ABORT if $f(\mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_1)) = y'$. Via an identical argument to Lemma 2, we can rely on the one-wayness of f to argue that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.
- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we generate fct_2 as $\mathsf{FHE.Enc}(\mathsf{fpk}_2, x')$. Via an identical argument to Lemma 3, we can rely on the security of the FHE scheme to to show that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.
- $\underline{\mathsf{Hyb}_6}$: In this hybrid, we compute $\widetilde{\mathbf{CC}}_2$ as $\mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x])$ where $u_2 \leftarrow \{0,1\}^\lambda$. Via an identical argument to Lemma 4, we can rely on the security of the compute and compare obfuscation to show that $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.
- $\underline{\mathsf{Hyb}_7}$: In this hybrid, we compute ct_2 as $\mathsf{RSR.Enc}(\mathsf{pk}_2, u_2)$. Via an identical argument to Lemma 5, we can rely on the security of the RSR encryption to show that $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$.
- $\underline{\mathsf{Hyb}_8}$: In this hybrid, we make the following changes:
  1. We compute $(\mathsf{pk}_2, \mathsf{sk}_2) = \mathsf{RSR.Gen}(1^\lambda; r_1)$ where $r_1 \leftarrow \{0,1\}^\lambda$.
  2. We compute $y' = f(x')$ where $x' \leftarrow \{0,1\}^\lambda$.
  3. We compute $\mathsf{ct}_2 = \mathsf{RSR.Enc}(\mathsf{pk}_2, u_2; r_2)$ where $r_2, u_2 \leftarrow \{0,1\}^\lambda$.
  4. We sample $(\mathsf{fpk}_2, \mathsf{fsk}_2) = \mathsf{FHE.Gen}(1^\lambda; r_3)$
  5. We compute $\mathsf{fct}_2 = \mathsf{FHE.Enc}(\mathsf{fpk}_2, x'; r_4)$
  6. We compute $\widetilde{\mathbf{CC}}_2 = \mathcal{O}(\mathbf{CC}[\mathsf{FHE.Dec}(\mathsf{fsk}_2, \cdot), u_2, x]; r_5)$
  7. We generate $\overline{\pi} \leftarrow \overline{\mathsf{ZAP.Prove}}(\overline{r}, \overline{z}, (x, u_2, x', r_1, r_2, r_3, r_4, r_5))$.

  Via an identical argument to Lemma 6, we can rely on the witness indistinguishability of $(\overline{\mathsf{ZAP.Prove}}, \overline{\mathsf{ZAP.Verify}})$ to show that $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$.

– $\underline{\mathsf{Hyb}_9}$: In this hybrid, we generate ct_1 as $\mathsf{RSR.Enc}(\mathsf{pk}_1, 0^\lambda)$ instead of $\mathsf{RSR.Enc}(\mathsf{pk}_1, u_1)$. Using a similar argument to Lemma 5, we can rely on the security of the RSR encryption to show that $\mathsf{Hyb}_8 \approx_c \mathsf{Hyb}_9$.
– $\underline{\mathsf{Hyb}_{10}}$: In this hybrid, we generate $\widetilde{\mathbf{CC}}_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|\mathsf{FHE.Dec}(\mathsf{fsk}_1,\cdot)|}, 1^\lambda)$. Using a similar argument to Lemma 4, we can rely on the security of the compute and compare obfuscation to show that $\mathsf{Hyb}_9 \approx_c \mathsf{Hyb}_{10}$.
– $\underline{\mathsf{Hyb}_{11}}$: In this hybrid, we generate fct_1 as $\mathsf{FHE.Enc}(\mathsf{fpk}_1, 0^\lambda)$. Via a similar argument to Lemma 3, we can rely on the security of FHE encryption to show that $\mathsf{Hyb}_{10} \approx_c \mathsf{Hyb}_{11}$.
– $\underline{\mathsf{Hyb}_{12}}$: In this hybrid, we generate com as $\mathsf{Com}(1^\lambda, 1)$. From the hiding property of Com, it follows that $\mathsf{Hyb}_{11} \approx_c \mathsf{Hyb}_{12}$.
– $\underline{\mathsf{Hyb}_{13}}$: In this hybrid, if $\mathsf{ZAP.Verify}(s, z, \pi) = 1$, then
  1. We compute $(\mathsf{sfe}_2, x_1) = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_3)$.
  2. We compute $\sigma := g(x_1, x_2)$ instead of using out.

  In Lemma 14, we rely on the soundness of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ to show that Hyb_{12} and Hyb_{13} are statistically indistinguishable.
– $\underline{\mathsf{Hyb}_{14}}$: In this hybrid, we make the following changes:
  1. We generate sfe_1 as $\mathsf{SFE}_1(1^\lambda, 0^\lambda)$ instead of $\mathsf{SFE}_1(1^\lambda, x_2)$.

  In Lemma 15, we rely on the receiver security of the SFE protocol to show that Hyb_{13} and Hyb_{14} are computationally indistinguishable.
– $\underline{\mathsf{Hyb}_{15}}$: In this hybrid, we again make the same changes as in Hyb_1 wherein we abort if $z' \notin L'$. Again, via an identical argument to Lemma 12, we can rely on the adaptive computational soundness and show that Hyb_{14} and Hyb_{15} are computationally indistinguishable.
– $\underline{\mathsf{Hyb}_{16}}$: In this hybrid, we consider a rewind thread where we generate the messages in the protocol as in Hyb_{15}. If we abort during this execution, we output the view of the adversary and instruct the ideal functionality to output $\perp$ to the honest receiver. If we have not aborted, we estimate the probability of not aborting. Specifically, we fix the first round message and generate second round messages until we get 12λ executions where the receiver has not aborted. Let m be the number of total number of trials and we set $\widetilde{\varepsilon} = 12\lambda/m$. Now, we repeatedly try to obtain an accepting transcript (called as the main thread) for $\lambda^2/\widetilde{\varepsilon}$ independent trials. If in each of the trail, we fail to obtain a valid transcript, we output a special symbol ABORT. Otherwise, we output the view of the adversary in the first valid transcript and instruct the ideal functionality to deliver the output of the ideal functionality to the honest receiver.

  Via a standard argument (see for instance the one given in [GK96]), we can show that Hyb_{15} and Hyb_{16} are statistically close.
– $\underline{\mathsf{Hyb}_{17}}$: In this hybrid, if the transcript is accepting in the first rewind thread, then we run the rewinding extractor for ExtCom as well as the trapdoor protocol to obtain s_1, x. We use this to generate the messages in the main thread. Since in the first rewind thread, we abort if $z' \notin L'$, it follows that the output of the rewinding extractor for ExtCom is identical to the non-uniformly fixed s_1. Again, from the property of the rewinding extractor for the trapdoor generation protocol, it follows that x is a valid trapdoor.
– $\underline{\mathsf{Hyb}_{18}}$: In this hybrid, we reverse the changes made in Hyb_{15} and again via an identical argument as before, it follows that Hyb_{17} and Hyb_{18} are computationally indistinguishable.
– $\underline{\mathsf{Hyb}_{19}}$: In this hybrid, we switch the messages generated in all the rewind threads as in the real execution of the protocol. Via an identical argument that showed indistinguishability between Hyb_0 and Hyb_{14}, it follows that Hyb_{18} is computationally indistinguishable to Hyb_{18}. We note that Hyb_{19} is identical to the output of the ideal experiment using the simulator.

Lemma 12. *Assuming the adaptive computational soundness of* $(\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2)$*, we have* $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$*.*

Proof. Assume for the sake of contradiction that Hyb_0 and Hyb_1 are computationally distinguishable with non-negligible advantage $\mu(\lambda)$. We now show that this contradicts the adaptive computational soundness of $(\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2)$.

Notice that the only difference between Hyb_0 and Hyb_1 is that in Hyb_1 if $z' \notin L'$ then we abort even if $\mathsf{wZK.V}_2$ accepts. Thus, if Hyb_0 and Hyb_1 are computationally distinguishable with advantage $\mu(\lambda)$, then it follows that the event $\mathsf{wZK.V}_2$ accepts and $z' \notin L'$ happens with probability $\mu(\lambda)$. We show that this contradicts the adaptive computational soundness.

We interact with the challenger for the adaptive computational soundness game and forward the messages corresponding to the weak zero-knowledge protocol received from the adversarial signer to the external challenger. After obtaining the final round message, we output z' along with wzk_3.

Note that the event $z' \notin L'$ and $\mathsf{wZK.V}_2$ accepts the proof happens with probability $\mu(\lambda)$ and this is non-negligible. Thus, the above described prover breaks the adaptive computational soundness with advantage $\mu(\lambda)$ and this is a contradiction.

Lemma 13. $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$.

Proof. Note that in Hyb_1, we abort whenever $z' \notin L'$. Thus, whenever $z' \in L'$, computing sfe_2 as $\mathsf{SKDec}(\mathsf{RSR.Dec}(\mathsf{sk}_1, \mathsf{RSR.Dec}(\mathsf{sk}_2, \overline{\mathsf{ct}})), \widehat{\mathsf{ct}})$ and computing it as $\mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}')$ are equivalent. Finally, it follows from the perfect correctness of encryption that if there exists $s_3, s_4 \in \{0,1\}^\lambda$ such that $\overline{\mathsf{ct}} = \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}, \bot; s_4); s_5)$ then we abort in Hyb_1 as well.

Lemma 14. $\mathsf{Hyb}_{12} \approx_s \mathsf{Hyb}_{13}$.

Proof. We first argue that in Hyb_{12}, if $\mathsf{ZAP.Verify}(s, z, \pi) = 1$ then the instance $z \in L_3$ with overwhelming probability. Suppose $z \notin L_3$ happens with non-negligible probability $\mu(\lambda)$. We show that this contradicts the soundness of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$.

Note that in Hyb_{12}, we have:

1. $(s_4, s_5) = \mathsf{Den.Dec}(\mathsf{sk}', \mathsf{ct}'_2)$ and $\overline{\mathsf{ct}} \neq \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \bot; s_4); s_5)$.
2. $f(\mathsf{Den.Dec}(\mathsf{sk}'.\mathsf{ct}'_1)) \neq y'$.
3. $\mathsf{com} = \mathsf{Com}(1^\lambda, 1)$

It now follows from the perfect correctness of $\mathsf{Den.Dec}$ and perfect binding of Com that $z \notin L_1 \vee L_2 \vee L_4$. Thus, if $z \notin L_3$ and if if $\mathsf{ZAP.Verify}(r, z, \pi) = 1$, then this contradicts the soundness of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$.

Note that if $z \in L_3$, then sfe_2 is correctly generated and x_1 is such that $\mathsf{sfe}_2 = \mathsf{SFE}_2(\mathsf{sfe}_1, g(x_1, \cdot))$. Thus, from the perfect correctness of SFE, we have $\sigma := g(x_1, x_2)$ in Hyb_{13} is equivalent to its computation in Hyb_{12}.

Lemma 15. *Assuming the receiver security of the SFE protocol, we have* $\mathsf{Hyb}_{13} \approx_c \mathsf{Hyb}_{14}$.

Proof. Note that the only difference between Hyb_{13} and Hyb_{14} is that in Hyb_{14} sfe_1 is generated as $\mathsf{SFE}_1(1^\lambda, 0^\lambda)$ whereas in Hyb_{13} it is generated as $\mathsf{SFE}_1(1^\lambda, x_2)$. Thus, it follows directly from the receiver security of the SFE protocol that $\mathsf{Hyb}_{13} \approx_c \mathsf{Hyb}_{14}$.

5.6 Receiver is Corrupt

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a malicious adversary that is corrupting the receiver. In the following, let p_i be the probability that the distinguisher outputs 1 in Hyb_i. Let $\varepsilon(\lambda)$ be the distinguishing parameter. The description of Sim_R is similar to the simulator for the weak zero-knowledge property and thus, we directly move to the proof of indistinguishability. The final hybrid experiment corresponds to the output of the ideal experiment using the simulator Sim_R (that is implicitly defined in the description of the hybrid.)

- $\underline{\mathsf{REAL}}$: This corresponds to the output of the adversarial receiver in the real execution of the protocol.
- $\underline{\mathsf{Hyb}_0}$: In this hybrid, we make the following changes:
  1. We generate $(\mathsf{wzk}_1, \mathsf{wzk}_2)$ using the simulator for weak zero-knowledge $\mathsf{Sim}_{\mathsf{wzk}}$ with 1^λ as the security parameter, $\mathcal{A}_1$ as the adversarial verifier, $\mathcal{A}_2$ be the distinguisher and $1^{10/\varepsilon}$ as the distinguishing parameter.

  In Lemma 16, we show that $|p_0 - p_{\mathsf{REAL}}| \leq \varepsilon/10$ from the weak zero-knowledge property of $(\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2)$.

– $\underline{\mathsf{Hyb}_2}$: In this hybrid, instead of generating $\mathsf{ECom}_1, \mathsf{ECom}_3$ as an extractable commitment to s_1, we generate it as an extractable commitment to a dummy message. It now directly follows from the hiding of the extractable commitment that $|p_1 - p_0| \le \mathsf{negl}(\lambda)$.

– $\underline{\mathsf{Hyb}_2}$: In this hybrid, we receive the second round message from $\mathcal{A}$ and parse it as $(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{fpk}_1, \mathsf{pk}_2, \mathsf{ct}_2, \mathsf{fpk}_2, y', \mathsf{sfe}_1, \mathsf{fct}_1, \mathsf{fct}_2, \widetilde{\mathbf{CC}}_1, \widetilde{\mathbf{CC}}_2, s, \mathsf{wzk}_2, \overline{\pi}, \mathsf{td}_2, \mathsf{ECom}_2)$. We construct a homomorphic simulation circuit $\mathsf{HS}_2[s'_1]$ as described in Figure 8.

We parse fct_2 as $(\mathsf{fct}_{2,1}, \ldots, \mathsf{fct}_{2,\lambda})$.
  - For each $k \in [\lambda]$, we compute $\overline{\mathsf{fct}}_{2,k} = \mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{Den.Enc}(\mathsf{pk}', \cdot; s'_{1,k}), \mathsf{fct}_{2,k}))$.
  - We compute $\mathsf{Sanitize}(\mathsf{fpk}_2, \mathsf{FHE.Eval}(\mathsf{fpk}_2, \mathsf{HS}_2[s'_1], (\overline{\mathsf{fct}}_{2,1}, \ldots, \overline{\mathsf{fct}}_{2,\lambda})) = \mathsf{fct}'$.
  - We compute $\widetilde{\mathbf{CC}}_2(\mathsf{fct}') = x$.

If $\mathsf{TDVerify}(\mathsf{td}_1, x) = 1$, then we abort. Via an identical argument to Lemma 8, we can show that $|p_1 - p_2| \le \mathsf{negl}(\lambda)$ using the soundness of the trapdoor generation protocol.

– $\underline{\mathsf{Hyb}_3}$: In this hybrid, we make the following changes. We construct a homomorphic simulation circuit HS_1 described in Figure 9.

We compute $\mathsf{FHE.Eval}(\mathsf{fpk}_1, \mathsf{HS}_1, \mathsf{fct}_1) = \mathsf{fct}'$. We run $\widetilde{\mathbf{CC}}_1(\mathsf{fct}') = (x_2, r_1, \rho)$. We check if $\mathsf{com} = \mathsf{Com}(1^\lambda, 0; \rho)$ and if $\mathsf{sfe}_1 \leftarrow \mathsf{SFE}_1(1^\lambda, x_2; r_1)$. If yes, we make the following changes:
  1. $\underline{\mathsf{Hyb}_{2,1}}$: We generate π as $\mathsf{ZAP.Prove}(s, z, \rho)$. Via an identical argument given in Lemma 9, we can show that using the witness indistinguishability of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ that $|p_2 - p_{2,1}| \le \mathsf{negl}(\lambda)$.
  2. $\underline{\mathsf{Hyb}_{2,2}}$: We generate ct' as $\mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$. Via an identical argument to Lemma 10, we can show that using the security of $\mathsf{Den.Enc}$ that $|p_{2,2} - p_{2,1}| \le \mathsf{negl}(\lambda)$.
  3. $\underline{\mathsf{Hyb}_{2,3}}$: We generate sfe_2 as $\mathsf{Sim}_{\mathsf{SFE}}(1^\lambda, x_2, r_1, g(x_1, x_2))$. In Lemma 17, we rely on the sender security of SFE to show that $|p_{2,3} - p_{2,2}| \le \mathsf{negl}(\lambda)$. Note that in this hybrid, $g(x_1, x_2)$ is identically distributed to the output of ideal functionality.

– $\underline{\mathsf{Hyb}_4}$: If
  - $\mathsf{TDVerify}(x, \mathsf{td}_1) \ne 1$, and
  - $\mathsf{com} \ne \mathsf{Com}(1^\lambda, 0; \rho)$ or $\mathsf{sfe}_1 \ne \mathsf{SFE}_1(1^\lambda, x_2; r_1)$

then we make the following changes:
  1. $\underline{\mathsf{Hyb}_{3,1}}$: We switch $\overline{\mathsf{ct}}$ generated as part of the final round message to $\mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{RSR.Enc}(\mathsf{pk}_1, \bot; s'_1); s'_2)$ where $s'_1, s'_2 \leftarrow \{0,1\}^\lambda$. Via an identical argument to Lemma 11, we show that $|p_{3,1} - p_3| \le 4\varepsilon/5 + \mathsf{negl}(\lambda)$
  2. $\underline{\mathsf{Hyb}_{3,2}}$: We generate ct'_2 in in the final round message as $\mathsf{Den.Enc}(\mathsf{pk}', (s'_1, s'_2); s'_3)$ where $s'_3 \leftarrow \{0,1\}^\lambda$. Via an identical argument to Lemma 10, we can use the security of $\mathsf{Den.Enc}$ to show that $|p_{i-1,2,2} - p_{i-1,2,1}| \le \mathsf{negl}(\lambda)$.
  3. $\underline{\mathsf{Hyb}_{3,3}}$: We generate $\pi \leftarrow \mathsf{ZAP.Prove}(s, z, (s'_1, s'_2, s'_3))$ (where π is part of $\mathsf{bs}_{2,i}$). Via an identical argument to Lemma 9, we can rely on the witness indistinguishability of $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ to prove that $|p_{3,2} - p_{3,3}| \le \mathsf{negl}(\lambda)$.
  4. $\underline{\mathsf{Hyb}_{3,4}}$: We generate $\mathsf{ct}' \leftarrow \mathsf{Den.Enc}(\mathsf{pk}', \mathbf{0})$ where $\mathbf{0}$ is a default input. Again via an identical argument to Lemma 10, we can rely on the security of $\mathsf{Den.Enc}$ to show that $|p_{3,3} - p_{3,4}| \le \mathsf{negl}(\lambda)$.
  5. $\underline{\mathsf{Hyb}_{3,5}}$: In this hybrid, we generate $\widehat{\mathsf{ct}}$ as an encryption of some default message. It directly follows from the security of the secret-key encryption that $|p_{3,5} - p_{3,4}| \le \mathsf{negl}(\lambda)$.

Note that $\mathsf{Hyb}_{3,5}$ is identically distributed to the output of the ideal experiment with the simulator.
From the above argument, we infer that $|p_{\mathsf{IDEAL}} - p_{\mathsf{REAL}}| \le \varepsilon/10 + 4\varepsilon/5 + \mathsf{negl}(\lambda) \le \varepsilon$.

Lemma 16. *Assuming the weak zero-knowledge property of* $(\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2)$*, we have that* $|p_{\mathsf{REAL}} - p_0| \le \varepsilon/10$

Proof. We construct a malicious verifier V^* that receives the first round message from the external challenger. It runs $V^* = \mathcal{A}_1$ on this message and obtains wzk_2 which it forwards to the external challenger. It receives the response wzk_3 from the challenger and uses it to generate the final round message in the protocol. Now,

– **Hardcoded:** s_1' and the transcript in the first two rounds and the third round message in the weak zero-knowledge sub-protocol.
– **Input:** ct_1'.
1. It recovers the message x' from ct_1' using pk' as the public key and s_1' as the randomness.
2. It constructs a distinguisher D_2 that takes pk_2 and $\overline{\mathsf{ct}}$ as input where $\overline{\mathsf{ct}}$ is either an encryption under the public key pk_2 of $\mathsf{RSR.Enc}(\mathsf{pk}_1, \mathbf{0})$ or $\mathsf{RSR.Enc}(\mathsf{pk}_1, \perp)$. The distinguisher D_2 generates π using x', s_1' as the witness. It generates the rest of the messages (using independently sampled x_1') in the third round as before except that it uses the provided $\overline{\mathsf{ct}}$. D_2 finally runs the $\mathcal{A}_2$ on the view of $\mathcal{A}$ and finally output whatever $\mathcal{A}_2$ outputs.
3. It then runs $\mathsf{RSR}.\widetilde{\mathsf{Dec}}^{D_2}(\mathsf{pk}_2, \mathsf{ct}_2, 1^{5/\varepsilon})$ to obtain u_2 and outputs u_2.

Figure 8: Description of HS_2

– **Hardcoded:** Transcript of the first two rounds of the protocol and the third round message in the weak zero-knowledge sub-protocol.
– **Input:** ρ.
1. It constructs a distinguisher D_1 takes as input $(\mathsf{pk}_1, \mathsf{ct}'')$ and ct'' is an RSR encryption under pk_1 of either sfe_2 or $\perp$. It generates $\overline{\mathsf{ct}} = \mathsf{RSR.Enc}(\mathsf{pk}_2, \mathsf{ct}'')$. It generates π using the witness ρ. It generates the rest of the messages (using independently sampled x_1') as in the protocol description and gives this to $\mathcal{A}$. It finally runs the distinguisher $\mathcal{A}_2$ on the view of $\mathcal{A}$ and outputs whatever it outputs.
2. It then computes $u_1 = \mathsf{RSR}.\widetilde{\mathsf{Dec}}^{D_1}(\mathsf{pk}_1, \mathsf{ct}_1, 1^{5q/\varepsilon})$ and outputs it.

Figure 9: Description of HS_1

run the distinguisher $D = \mathcal{A}_2$ on the view of the adversary $\mathcal{A}$ and output whatever it outputs. We consider an $\mathsf{InstGen}$ that outputs the instance z' along with the witness $(\mathsf{sfe}_2, s_1, s_2, s_4, s_5)$.

Note that by the above definition p_{REAL} corresponds to the probability that $\mathsf{REAL}(1^\lambda, V^*, D, q, \mathsf{InstGen}) = 1$ and p_0 corresponds to the probability that $\mathsf{IDEAL}(1^\lambda, V^*, D, q, \mathsf{Sim}_{\mathsf{wzk}}, 1^{10/\varepsilon}, \mathsf{InstGen}) = 1$. Hence, $|p_{\mathsf{REAL}} - p_0| \leq \varepsilon/10$ from the weak zero-knowledge property of $(\mathsf{wZK.P}_1, \mathsf{wZK.P}_2, \mathsf{wZK.V}_1, \mathsf{wZK.V}_2)$.

Lemma 17. *Assuming the sender security of SFE protocol, we have* $|p_{2,3} - p_{2,2}| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there is a non-negligible function $\mu(\cdot)$ such that $|p_{2,2} - p_{2,3}| \geq \mu(\lambda)$. We now show that this contradicts the sender security of the SFE protocol.

Note that the only difference in $\mathsf{Hyb}_{2,2}$ and in $\mathsf{Hyb}_{2,3}$ is in how sfe_2 is generated. In $\mathsf{Hyb}_{2,2}$, it is generated using the honest sender algorithm SFE_2 whereas in $\mathsf{Hyb}_{2,3}$, it is generated as $\mathsf{Sim}_{\mathsf{SFE}}(1^\lambda, x_2, r_1, g(x_1, x_2))$. Thus, if $|p_{2,2} - p_{2,3}| \geq \mu(\lambda)$, then the sender security of SFE protocol does not hold and this is a contradiction.

References

ABG+21. Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. Two-round maliciously secure computation with super-polynomial simulation. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 654–685. Springer, 2021.

AIR01. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, Heidelberg, May 2001.

AJ17. Prabhanjan Ananth and Abhishek Jain. On secure two-party computation in three rounds. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 612–644. Springer, Heidelberg, November 2017.

Bar01. Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

BD18. Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In *TCC 2018, Part II*, LNCS, pages 370–390. Springer, Heidelberg, March 2018.

BFJ+20. Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In Anne Canteaut and Yuval Ishai, editors, *Eurocrypt*, volume 12107, pages 642–667. Springer, 2020.

BGGL01. Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *42nd FOCS*, pages 116–125. IEEE Computer Society Press, October 2001.

BGJ+17. Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 743–775. Springer, Heidelberg, November 2017.

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. LNCS, pages 459–487. Springer, Heidelberg, 2018.

BKP18. Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In *50th ACM STOC*, pages 671–684. ACM Press, 2018.

BKP19. Nir Bitansky, Dakshita Khurana, and Omer Paneth. Weak zero-knowledge beyond the black-box barrier. In *51st ACM STOC*, pages 1091–1102. ACM Press, 2019.

BL02. Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th ACM STOC*, pages 484–493. ACM Press, May 2002.

BM82. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.

BP12. Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *53rd FOCS*, pages 223–232. IEEE Computer Society Press, October 2012.

BP13. Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 241–250. ACM Press, June 2013.

BS05. Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *46th FOCS*, pages 543–552. IEEE Computer Society Press, October 2005.

CCL18. Yi-Hsiu Chen, Kai-Min Chung, and Jyun-Jie Liao. On the complexity of simulating auxiliary input. LNCS, pages 371–390. Springer, Heidelberg, 2018.

CLP13. Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from P-certificates. In *54th FOCS*, pages 50–59. IEEE Computer Society Press, October 2013.

CLP15. Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 287–307. Springer, Heidelberg, August 2015.

DGI+19. Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. LNCS, pages 3–32. Springer, Heidelberg, 2019.

DGS09. Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *50th FOCS*, pages 251–260. IEEE Computer Society Press, October 2009.

DN00. Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, November 2000.

DNRS99. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, October 1999.

DS16. Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 294–310. Springer, Heidelberg, May 2016.

ElG86. Taher ElGamal. On computing logarithms over finite fields. In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 396–402. Springer, Heidelberg, August 1986.

FLS90. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990.

GJJM20. Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical zaps and new oblivious transfer protocols. In Anne Canteaut and Yuval Ishai, editors, *Eurocrypt*, volume 12107, pages 668–699. Springer, 2020.

GK96. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptol.*, 9(3):167–190, 1996.

GKVW20. Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC*, volume 12550, pages 229–259. Springer, 2020.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *58th FOCS*, pages 612–621. IEEE Computer Society Press, 2017.

GM82. Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

GM11. Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 678–687. IEEE Computer Society Press, October 2011.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GOS12. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

Goy13. Vipul Goyal. Non-black-box simulation in the fully concurrent setting. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 221–230. ACM Press, June 2013.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

GS09. Vipul Goyal and Amit Sahai. Resettably secure computation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 54–71. Springer, Heidelberg, April 2009.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

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

IKO+11. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

JP14. Dimitar Jetchev and Krzysztof Pietrzak. How to fake auxiliary input. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 566–590. Springer, Heidelberg, February 2014.

Kal05. Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, Heidelberg, May 2005.

KO04. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004.

LVW20. Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Statistical ZAPR arguments from bilinear maps. In Anne Canteaut and Yuval Ishai, editors, *Eurocrypt*, volume 12107 of *Lecture Notes in Computer Science*, pages 620–641. Springer, 2020.

NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 448–457. ACM/SIAM, 2001.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

Pas03. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003.

Pas04. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th ACM STOC*, pages 232–241. ACM Press, June 2004.

PR03. Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *44th FOCS*, pages 404–415. IEEE Computer Society Press, October 2003.

PR05. Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *46th FOCS*, pages 563–572. IEEE Computer Society Press, October 2005.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *58th FOCS*, pages 600–611. IEEE Computer Society Press, 2017.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.