On the Impossibility of Actively Secure Distributed Samplers

Damiano Abram[1], Serge Fehr[2], Maciej Obremski[3], and Peter Scholl[4]

[1] University of Edinburgh[**]
[2] CWI Amsterdam and Leiden University
[3] National University of Singapore
[4] Aarhus University

Abstract. One-round secure computation is generally believed impossible due to the *residual function attack*: any honest-but-curious participant can replay the protocol in their head changing their input, and learn, in this way, a new output. Inputless functionalities are among the few that are immune to this problem.
This paper studies one-round, multi-party computation protocols (MPC) that implement the most natural inputless functionality: one that generates a random sample from a fixed distribution. These are called *distributed samplers.* At Eurocrypt 2022, Abram, Scholl and Yakoubov showed how to build this primitive in the semi-honest model with dishonest majority. In this work, we give a lower bound for constructing distributed samplers with a malicious adversary in the standard model. More in detail, we show that for any construction in the stand-alone model with black-box simulation, even with a CRS and honest majority, the output of the sampling protocol must have low entropy. This essentially implies that this type of construction is useless in applications. Our proof is based on an entropic argument, drawing a new connection between computationally secure MPC, information theory and learning theory.

1 Introduction

Even in the semi-honest model, any one-round computation protocol is subject to the residual function attack [HIJ+17]: any participant can replay the protocol in their head, reusing the other parties' messages in combination with a new input. From any such virtual execution, the attacker learns a new output. This is fatal for most applications, as the freshly derived outputs often leak information about the other parties' inputs. There is, however, one situation that is immune to the attack: *no-signaling* functionalities [DLN+04, DHRW16]. These are functionalities where the outputs of any subset of corrupted parties look independent of the remaining parties' inputs. For instance, any inputless functionality is no-signaling.

This paper studies one-round protocols that implement the most natural, non-trivial inputless functionality: one that generates a random sample from a fixed distribution $\mathcal{D}$. This question is also motivated from a real-word perspective. Many cryptographic protocols require public parameters to be generated in a secure manner. This is the case, for instance, with trusted parameters used in many succinct zero-knowledge proofs [BCCT12], or trusted RSA moduli used in cryptographic accumulators [Bd94]. These parameters often hide *trapdoors*, artifacts of the security proofs that, if leaked to an attacker, could completely compromise the desired security properties. For this reason, it is important that the generation of these parameters is performed correctly and securely.

A common practice is to delegate the generation of this sensitive material to trusted authorities, for instance a public institution or a private corporation: essentially, due to their reputation and accountability, we have confidence that they generate the public parameters correctly and ensure that all trapdoors remain secret. Ignoring the fact that it is debatable whether we can really trust that these institutions will not take advantage of their privileged position (remember the Snowden files), this practice inherently introduces a point of failure in the constructions: what if a cyber attack manages to breach into the servers of the trusted institution? What if they succeed in learning any of the trapdoors?

As a result, when such parameters are needed, the parties involved may wish to run a secure multiparty computation protocol to generate them, guaranteeing security and secrecy of the trapdoors as long as sufficiently many parties are honest. However, this type of setup protocol is typically expensive to carry out and coordinate. This is especially true for small end-users, normal people around the globe that only rarely need to run secure computations and, consequently, cannot easily amortise the costs of these setups.

[**] Work done while the authors was affiliated with Aarhus University and Bocconi University.

Security model	Setup	Corruption Threshold	Feasibility
Semi-malicious	None	Dishonest majority	poly iO + OWF [ASY22]
Malicious, UC	pRO	Dishonest majority	poly iO + OWF [ASY22]
Malicious, game	CRS	Dishonest majority	subexp iO + subexp OWF + ELF [AWZ23]
Malicious, UC	None	Dishonest majority	impossible [CKL03]
Malicious, BB	None	Dishonest majority	impossible [HV16, PVW08]
Malicious, BB	CRS	Any $t \geq \epsilon \cdot n$	impossible (§**4**)

Table 1: Overview of the feasibility of distributed samplers in different settings. UC = universal composability; BB = stand-alone with black-box simulation; game=game-based security definition; pRO = programmable random oracle; poly/subexp = polynomial/subexponential hardness; ELF = extremely lossy function. Observe that, since our results hold in a weaker setting (i.e. honest majority with CRS and black-box simulation), our impossibility is stronger.

Distributed samplers, recently introduced by Abram, Scholl and Yakoubov [ASY22, AWZ23], propose a solution to these issues by allowing parameters to be sampled using a *one-round* protocol: each party publishes a single message, after which everybody can obtain the same random sample from the desired distribution. More formally, a distributed sampler for n parties and a distribution $\mathcal{D}$ is defined by a pair of algorithms $(\mathsf{Gen}, \mathsf{Sample})$, such that given a set of messages $U_i = \mathsf{Gen}(1\!\!1^\lambda, i)$, for $i \in [n]$, one can compute a sample $R \leftarrow \mathsf{Sample}(U_1, \ldots, U_n)$[5]. The security requirement essentially states that this one-round protocol must securely realize the ideal functionality for sampling from $\mathcal{D}$.

In their work, Abram, Scholl and Yakoubov studied this primitive in the dishonest majority setting, both with passive and active security [ASY22]. Active security is particularly challenging, due to the need to handle a rushing adversary, who may choose their messages U_i *after* seeing the messages U_j of the other parties. This allows an attacker to "grind" different choices of their randomness, obtaining different U_i, until finding an output R that she likes. So, the best form of security one can hope for in this setting is a relaxation of the ideal functionality for sampling, where the adversary first obtains several samples from $\mathcal{D}$, before settling on a final output.

Abram et al [ASY22] constructed distributed samplers in the plain model (no CRS) for any distribution based on indistinguishability obfuscation. Their first construction is secure only against a *semi-malicious*[6] and non-rushing adversary. This was then upgraded to malicious security in the programmable random oracle model. On top of this, they showed how distributed samplers can be used for sampling arbitrary forms of *correlated randomness*, often used in MPC protocols, with a one-round protocol.

1.1 Our Results

In this work, we further explore the feasibility of distributed samplers. See Table 1 for an overview of our results and prior relevant work.

Impossibility of Distributed Samplers Without Random Oracles. We first pose the question: is it possible to build actively secure, one-time distributed samplers in the *standard model*, that is, without random oracles? As a starting point, we observe that actively secure distributed samplers cannot be built without a common reference string (CRS) in the UC model [Can01]. This is an immediate consequence of the UC impossibility for same-output probabilistic functions of [CKL03], since the function $\mathcal{D}(1\!\!1^\lambda)$ we want to compute has an unpredictable output and no inputs.

We observe also that generic actively secure distributed samplers without a CRS cannot exist even in the standalone model with black-box simulation. If that was not the case, by sequentially composing a distributed sampler with 2-round active OT protocols in the CRS model such as [PVW08], we would obtain a 3-round OT protocol with active security and black-box simulation in the plain model. The latter is known to be impossible [HV16].

[5] λ represents the security parameter.

[6] A semi-malicious adversary is one who follows the protocol, but may choose their random tape arbitrarily.

For this reason, we investigate the CRS model. At first glance, it seems that distributed samplers are then trivial: the CRS can directly encode a sample from the desired distribution. This solution does not even need interaction, however, interactive distributed samplers with a CRS may have some advantages over this trivial solution: the CRS could be reused multiple times, or it could be easier to generate, by being short or unstructured (i.e. a uniformly random string). We prove that, if the distributed sampler is secure against active adversaries in the stand-alone model with black-box simulation, none of the above properties can be satisfied, even if the adversary corrupts only $t = \epsilon \cdot n$ parties for any constant $\epsilon > 0$. We prove this by relying on a new kind of information-theoretic argument: by analysing the Shannon entropy diagram of the protocol, we show that any black-box simulator would fail to simulate even adversaries that follow the protocol but generate their randomness using a w-wise independent hash function (for a sufficiently large w) applied on the honest-parties messages. Essentially, at the base of the proof is the impossibility for any black-box simulator to learn the w-wise independent hash function, which would allow it to predict the adversary's randomness and simulate the protocol. Our argument therefore draws new connections between computationally secure MPC, information theory and learning theory.

All of these impossibilities come from our main result, below.

Theorem 1.1 (Informal, c.f. Thm. 4.1). *For any t-out-of-n distributed sampler that is secure against rushing adversaries in the stand-alone model with black-box simulation, we have*

$$\mathsf{H}(R \mid \sigma) = \frac{n}{t} \cdot O(\log \lambda),$$

where σ denotes the CRS and R the output of the distributed sampler.[7]

This essentially rules out this flavour of distributed sampler for all practical applications, as we discuss in the following corollaries.

Corollary 1: the collision probability is large. An immediate consequence of small Shannon entropy is that the output of the distributed sampler has a high probability of a collision if the CRS is not changed. This implies that in applications where more than one sample from $\mathcal{D}$ is needed, the same CRS cannot be reused.

Corollary 2: the CRS must be long. Less trivially, we show that this means that the CRS must be at most $O(\lambda)$ bits smaller than the Yao incompressibility entropy of $\mathcal{D}$. Recall that this roughly measures the compressed size of a sample from $\mathcal{D}$, after applying any efficient compression algorithm. As a result, the CRS must be almost as long as an output of $\mathcal{D}$, after applying compression.

Corollary 3: the CRS must be ugly. Finally, we show that in meaningful scenarios, the CRS must inherently be *structured*, or "ugly", meaning that it requires private coins to sample. In practice, this type of CRS must be generated by a trusted party or multi-party computation protocol, whereas obtaining a CRS that can be sampled from uniform randomness is much easier, relying only on a public source of randomness (or a hash function modeled as a random oracle).

Conclusion. Put together, the above corollaries show that, if there exists a $\omega(\log \lambda)$ multiplicative gap between the Shannon entropy of the distribution $\mathsf{H}(\mathcal{D})$ and the corruption ratio n/t, actively-secure distributed samplers with black-box simulation are essentially useless: given that the CRS can only be used once, is structured, and roughly as long as an output of $\mathcal{D}$, in practice it will most likely be no easier to generate the CRS than to just generate a sample from $\mathcal{D}$.

On the tightness of the lower bound. We observe that if $\mathsf{H}(\mathcal{D}) \leq n/t \cdot O(\log \lambda)$, it is sometimes possible to build distributed samplers with black-box simulation. For instance, suppose that $\mathcal{D}$ outputs a random string in $\{0,1\}^n$. Then, if $t = O(\log \lambda)$, we can build a black-box simulatable t-out-of-n distributed sampler for $\mathcal{D}$. The protocol is simple: each party P_i broadcasts a random bit b_i. The output consists of $(b_1, \ldots, b_n)$. To simulate, it is sufficient to query a random string x from the functionality and make all honest parties output a bit according to it. Given that there are at most $O(\log \lambda)$ corrupted parties, the bits chosen by the adversary will be consistent with x with $1/\mathsf{poly}(\lambda)$ probability. In the case of an unlucky event, the simulator can simply rewind and retry with another random string received from the functionality.

[7] We use H to denote the Shannon entropy. We refer to Section 3 for formal definitions.

How to interpret the impossibility. Our lower bound *must not* be read as a general impossibility for actively secure distributed samplers with non-trivial efficiency: the message we are trying to pass is that, if we want to construct such primitive, we need to rely on *non-black-box techniques* [Bar01]. As a matter of fact, the adversarial class for which we prove the impossibility of a black-box simulator is one for which we can easily build a non-black-box simulator. This is because for suitable w-wise independent hash functions, given an input-output pair (x, y), we can efficiently sample a random hash key that maps x to y. Therefore, to simulate the protocol, we can generate a transcript using a semi-honest simulator and then generate the hash key (the adversary's random tape) so that the honest messages are mapped to the simulated randomness for the adversary. Notice, however, that if we consider slightly more complex ways of generating the randomness for our honest adversary, the question of non-black-box simulation becomes immediately hard. For instance, what if we substitute the w-wise independent hash function with a PRF? What if we substitute it with a correlation-intractable hash function? Simulating any such adversary seems to intrinsically require the use of non-black-box techniques.

Notice that, even after years of research, our knowledge of the non-black-box world is still rather limited, and this is even more so in the context of non-interactive protocols. This seems to suggest that constructing non-trivial, actively secure distributed samplers is a particularly difficult task.

1.2 Related Work

Preliminary version. A preliminary, unpublished draft of this work can be found on the IACR ePrint Archive [AOS23]. The two versions of the work differ mainly in two aspects. First, the preliminary impossibility was proven for the UC model [Can01] with full-threshold corruption. In this submission, the impossibility has been *strengthened* to the standalone model with black-box simulation and a potentially honest majority. Secondly, in the preliminary draft, we explained how to build party-dynamic distributed samplers in the random oracle model. These additional results have been moved to another work.

Game-based actively secure distributed samplers. In [AWZ23], Abram, Waters and Zhandry presented solutions to circumvent the impossibility proven in this paper. Instead of aiming for a simulation-based security definition, they show that, using strong primitives (including subexponential iO) but no random oracle, it is possible to implement game-based definitions for distributed samplers that allow removing trusted setups in one round while preserving the hardness of search problems and the security of most protocols against active adversaries.

The lower bounds in this paper offer an argument that — notwithstanding a major advance in non-black-box simulation — the complex game-based definitions of [AWZ23] are necessary, as simulation-based security is unachievable without a random oracle. We point out that a simulation-based definition is, in principle, desirable as there exist situations for which the definitions of [AWZ23] are not sufficient. For instance, their notion of a hardness-preserving distributed sampler does not allow removing the CRS from a NIZK while preserving soundness. This is because hardness-preserving distributed samplers preserve the hardness of games only when the challenger is efficient. Their second notion of indistinguishability-preserving distributed samplers also does not work in all contexts. For example, consider the functionality $\mathcal{F}$ that provides the adversary with several RSA moduli, lets the adversary choose one of them (denote the chosen modulus by N), and then allows MPC over $\mathbb{Z}_N$ (this is an interesting setting, e.g. for using MPC tools from [OSY21] that require a trusted setup). There exists a protocol Π that, given an RSA modulus as CRS, implements $\mathcal{F}$. However, if we apply an indistinguishability-preserving distributed sampler [AWZ23] to generate the CRS, the result Π' cannot be proven to securely implement the functionality $\mathcal{F}$ using black-box techniques (following our main result).

Coin tossing extension. In [ADIN24], Abram et al proved a lower-bound for statistically secure coin-tossing extension with black-box simulation using an entropic argument similar to the one we used in Theorem 4.1. Their setting differs from ours in two important ways. First of all, unlike us, they work with statistically secure protocols. This significantly simplifies their argument given that Shannon entropy is preserved under statistical indistinguishability. On the other hand, Shannon entropy of computationally indistinguishable random variables can assume significantly different values. The other big difference is that their impossibility holds in the dishonest majority setting, whereas our lower-bound holds even in the presence of an honest majority (making our negative result stronger).

2 Technical Overview

We now give a high-level overview of the techniques used to obtain our results.

Notation. We denote the security parameter by λ. Even when not explicitly written, we assume that all random variables depend on λ. We use bold font to denote random variables. We use $\mathsf{Supp}(\mathbf{X})$ to denote the support of the random variable $\mathbf{X}$. The symbol $\sim_c$ denotes computational indistinguishability. We indicate the bit-length of any string s by $|s|$. With an abuse of notation, we identify distributions $\mathcal{D}$ with circuits mapping uniformly random strings of bits into samples. We say that a distribution is efficient if its circuit has $\mathsf{poly}(\lambda)$ size. We represent the number of parties by n, the set of corrupted players by C, the set of honest players by H. Throughout the whole paper, we assume that n is at most a polynomial quantity in the security parameter λ. We use $\mathsf{negl}(\lambda)$ to denote an arbitrary negligible function, i.e., $\mathsf{negl}(\lambda) = \lambda^{-\omega(1)}$. We say that an event E holds with overwhelming probability if $\Pr[\neg E] \leq 1 - \mathsf{negl}(\lambda)$.

2.1 (Im)possibility of Distributed Samplers without Random Oracle

As we motivated in the introduction, actively secure distributed samplers in the plain model with black-box simulation are impossible. In the CRS model, instead, they are trivial to build: the CRS can directly encode a sample from the underlying distribution. The result is a distributed sampler in which the parties do not even need to communicate, since they just output the CRS.

We study how interactive constructions can improve upon the trivial solution. In principle, the advantages can be multiple: the same CRS can be reused in many distributed sampler executions producing independent-looking outputs. Moreover, the CRS of distributed samplers can be nicer (i.e. easier to generate) than the direct encoding of a sample, for instance because of the smaller size, or because it is unstructured (i.e. a uniformly random string of bits). The result of our analysis is that none of the above properties can be satisfied: without a random oracle, distributed samplers with black-box simulation essentially provide no advantage over the trivial solution. In order for this impossibility to hold, we do not even need to aim for active security, it suffices that the adversary is *strongly semi-malicious*: it may adaptively choose the randomness of the corrupted parties after seeing the honest messages, but all corrupted players follow the protocol. Even more surprisingly, in the proof, we will show that no black-box simulator is able to simulate adversaries that produce the randomness of the corrupted parties deterministically, using a w-wise independent hash function for a sufficiently large w. Observe that all of these adversaries can actually be simulated very easily, but *not in a black-box manner*, intuitively, because the w-wise independent hash function is not learnable by a simulator with query complexity $< w$.

On the Unpredictability of Distributed Samplers in the CRS Model. All the negative results mentioned above are consequences of the main theorem of this work: in a strongly semi-malicious distributed sampler with black-box simulation, the Shannon entropy of the output conditioned on the CRS is $O(\log \lambda)$. Interestingly, this holds even in the honest majority setting when the corruption threshold is a constant fraction of the total number of participants.

Throughout the paper, we carefully juggle different variants of entropy, each bringing a unique set of properties we require during the proofs. Shannon entropy H has a powerful chain rule. Collision entropy H_2 gives us an elegant tool for building distinguishers, but lacks a chain rule and is not invariant under computational indistinguishability (i.e. for two computationally indistinguishable random variables, H_2 can be vastly different). Finally, Yao's entropy is the only entropy we use that remains invariant under computational indistinguishability (i.e. two computationally indistinguishable random variables have the same Yao entropy). For formal definitions please refer to Section 3.

Our definition of distributed samplers. To understand the idea behind the result, we first recall the definition of distributed samplers with security against an active adversary [ASY22]. We consider a classical simulation-based security definition. Specifically, we work in the stand-alone model with black-box simulation [GK90, Ore87, BMO90]: we require the existence of a single PPT simulator that, given oracle access to the adversary and the functionality, should be able to reproduce an adversarial view that is consistent with the functionality's outputs. In particular, we study *computational security*: the view of the adversary in the protocol should be computationally indistinguishable from the output of the simulator even if we leak the outputs of the honest parties.

The distributed sampler functionality provides the adversary with as many samples from the underlying distribution as the adversary wants. The adversary can then select one of these, which the functionality outputs to all the honest parties. This kind of behaviour is needed to model the fact that, in the case of a rushing adversary, the corrupted parties see the honest messages before they publish their own. In other words, before committing to a choice, they can always test their candidate messages and discard them if they are not happy. Below, we sketch the definition, a more formal version can be found in Section 4.

Definition 2.1 (Distributed sampler - informal security against rushing adversaries). *Let $\mathcal{D}(\mathbb{1}^\lambda)$ be an efficiently samplable distribution and let t be a corruption threshold. An n-party actively secure (resp. strongly semi-maliciously secure) distributed sampler for $\mathcal{D}(\mathbb{1}^\lambda)$ is a one-round protocol implementing the functionality $\mathcal{F}_\mathcal{D}$ (see Fig. 1) against a static and active (resp. strongly semi-malicious) adversary corrupting up to t parties.*

THE FUNCTIONALITY $\mathcal{F}_\mathcal{D}$

Initialisation. On input Init from every honest party and the adversary, the functionality activates and sets $Q := \emptyset$. (Q will be used to keep track of queries.) If all the parties are honest, the functionality outputs $R \xleftarrow{\$} \mathcal{D}(\mathbb{1}^\lambda)$ to every honest party and sends R to the adversary, then it halts.

Query. On input Query from the adversary, the functionality samples $R \xleftarrow{\$} \mathcal{D}(\mathbb{1}^\lambda)$ and creates a fresh label id. It sends (id, R) to the adversary and adds the pair to Q.

Output. On input $(\mathsf{Output}, \widehat{\mathsf{id}})$ from the adversary, the functionality retrieves the only pair $(\mathsf{id}, R) \in Q$ with $\mathsf{id} = \widehat{\mathsf{id}}$. Then, it outputs R to every honest party and terminates.

Fig. 1: The distributed sampler functionality for rushing adversaries

Our adversarial model. Our proof will only consider adversaries that behave almost completely honestly: the adversary will generate the corrupted parties' messages following the protocol execution, but will produce their randomness by running a w-wise independent hash function on the honest parties' messages and the CRS. Here, w denotes a polynomial upper bound on the number of times the simulator queries the adversary, so rewinding the adversary will give no advantage to the simulator. Notice that since we are proving a lower bound, considering very weak adversaries makes our results even stronger.

In the real world, the output is easily predictable from the CRS and the messages of the honest parties. Our starting goal is to show that, in the real world, $\mathsf{H}(R|U_H, \sigma) = O(\log \lambda)$, where R denotes the output of the distributed sampler, U_H denotes the set of messages of the honest parties and σ represents the CRS. We show that if that is not the case, we would be able to distinguish between the real and the ideal world with non-negligible advantage. The attack is very simple: the distinguisher estimates the probability that, by rerunning the protocol with the same CRS, the same messages for the honest parties, but fresh messages from the corrupted parties, the output is again R.

The intuition is that such probability is significantly higher in the ideal world. Since the adversary generates the messages of the corrupted parties using the w-wise independent hash function (which is not learnable), in order to make the simulated transcript consistent with the output, the simulator needs to test it by querying the candidate CRS and the honest messages to the adversary. This means that in at least one of the w queried executions, the adversary ended up choosing R as the output. Furthermore, this is not just a random R, it must be one of the ideal samples provided by the functionality! Notice that there are polynomially many of these, let's say at most q. If the simulator wants to be sure that, in at least one of the w tested executions, the adversary chooses one of these q ideal samples, it must be that these elements have a higher-than-usual probability of getting picked. As a consequence, the distribution of R must be biased towards elements that have a high probability of being resampled.

The final bound. To prove that $\mathsf{H}(R|\sigma) = \frac{n}{t} \cdot O(\log \lambda)$, we rely on the following lemma, proven in Section 4. We believe that this result could be of independent interest. Essentially, it states that, in a one round protocol, the influence of an adversary corrupting at most t parties, is proportional to the ratio t/n.

Lemma 2.1 (Informal). *Let $X_1, \dots, X_n, Y$ be random variables and suppose that $X_1, \dots, X_n$ are all independent conditioned on Y. Let $R \leftarrow f(X_1, \dots, X_n, Y)$ be a deterministic function. Then, for every $\tau \in [n]$, there exists a subset $S_\tau \subseteq [n]$ such that $|S_\tau| = \tau$ and*

$$\mathsf{H}(R|X_{S_\tau}, Y) \geq \frac{n-\tau}{n} \cdot \mathsf{H}(R|Y).$$

To see why this lemma implies our lower bound, observe that the output of the distributed sampler is a deterministic function of $U_1, \dots, U_n, \sigma$. Moreover, since our adversary is almost honest, in the real world, $U_1, \dots, U_n$ are all independent conditioned on σ. We conclude that there exists a subset H of $n-t$ elements such that

$$\frac{t}{n} \cdot \mathsf{H}(R|\sigma) \leq \mathsf{H}(R|U_H, \sigma) = O(\log \lambda).$$

Bad News for Distributed Samplers. All the results we discuss below hold in absence of a random oracle and for distributed samplers that achieve t-out-of-n security (for $t/n = \Theta(1)$) with black-box simulation against our class of almost-honest adversaries.

Distributed sampler CRSs cannot be used twice. Suppose that the distribution underlying the distributed sampler has high min-entropy, i.e. $\mathsf{H}_\infty(\mathcal{D}) = \omega(\log \lambda)$. We would hope that if we run the distributed sampler twice using the same CRS, the resulting outputs look like two independent samples from $\mathcal{D}(\mathbb{1}^\lambda)$ and therefore, they differ with overwhelming probability. Alas! This is impossible: the first corollary of Theorem 1.1 is that two distributed sampler executions using the same CRS have colliding outputs with non-negligible probability.

The reason at the base of our first corollary is that, by a simple application of Jensen's inequality, the average collision entropy $\widetilde{\mathsf{H}}_2(R|\,\sigma)$ is bounded from above by $\mathsf{H}(R|\,\sigma) = O(\log \lambda)$. We recall that the average collision entropy is defined as

$$\widetilde{\mathsf{H}}_2(R|\,\sigma) := -\log\big(\Pr[R = R']\big)$$

where R and R' are two distributed sampler outputs computed using the same CRS σ and the probability is also over the randomness of σ. We conclude that $\Pr[R = R'] \geq 1/\mathsf{poly}(\lambda)$.

Distributed sampler CRSs are long. We prove that CRSs of strongly semi-malicious distributed samplers cannot be small: they can be at most $O(\log \lambda)$ bits shorter than the Yao entropy of the underlying distribution $\mathsf{H}_{\mathsf{Yao}}(\mathcal{D})$[8].

We prove this result by first observing that $\mathsf{H}_{\mathsf{Yao}}(R|\,\sigma) = O(\log \lambda)$. Indeed, as we motivated in the previous paragraph, two distributed sampler executions using the same CRS have colliding outputs with non-negligible probability. We can therefore consider the Yao's compressor that outputs nothing and the associated decompressor that, provided with the CRS σ, reruns the distributed sampler protocol in its head and outputs the result R'. With $1/\mathsf{poly}(\lambda)$ probability, R' coincides with the input of the compressor.[9] This is enough to conclude that $\mathsf{H}_{\mathsf{Yao}}(R|\,\sigma) = O(\log \lambda)$.

We then show that $\mathsf{H}_{\mathsf{Yao}}(R|\,\sigma) \geq \mathsf{H}_{\mathsf{Yao}}(R) - |\sigma|$. We prove this by noticing that, given a compressor-decompressor pair (c', d') for $\mathsf{H}_{\mathsf{Yao}}(R|\,\sigma)$, we can build a compressor-decompressor pair (c, d) for $\mathsf{H}_{\mathsf{Yao}}(R)$ as follows: c provides its input R to the distributed sampler simulator, corrupting no party. It obtains a fake CRS σ' that looks like the real one. It then outputs $c'(R, \sigma')$ along with σ'. The decompressor d is exactly the same as d'. The success probability of (c, d) is the same as for (c', d') except for a negligible quantity. The size of the compressed string has however grown by $|\sigma|$ bits, increasing $\mathsf{H}_{\mathsf{Yao}}(R)$ by the same amount.

We point out that, in order to prove the above inequality, we cannot use the Yao chain rule of [KPW13, Appendix B] as their compressor for $\mathsf{H}_{\mathsf{Yao}}(R)$ has $O(2^{|\sigma|})$ size.

[8] The Yao entropy of $\mathcal{D}$ roughly measures how much a sample from $\mathcal{D}$ can be compressed in polynomial time without losing information.

[9] We can make the decompressor deterministic using a PRF.

Distributed sampler CRSs are ugly. Suppose that there exists a strongly semi-malicious distributed sampler for the distribution $\mathcal{D}$ having CRS σ. We prove that it is possible to *non-interactively* and *securely* generate a sample from $\mathcal{D}$ given only σ and public random coins. In other words, if there exists a distributed sampler with nice CRS, also the underlying distribution can be encoded in a nice CRS. The second solution may be preferable as it often requires less communication. As an additional corollary, if the distributed sampler uses a URS (i.e. the CRS is a random string of bits), we can sample from $\mathcal{D}$ using just public random coins. So, in the random oracle model, we would not even need a CRS.

Our idea is that, given σ and public random coins, each party can just rerun the distributed sampler protocol with σ as CRS and the public coins as randomness for the players. The distribution result R is clearly indistinguishable from a sample from $\mathcal{D}$. However, in order to prove that this protocol is secure, we need to be able to simulate σ and the public coins, given R.

We simulate σ by feeding R to the distributed sampler simulator (we corrupt no party). Unfortunately, the simulator cannot provide us with the randomness used by the parties. We proceed by brute-force: we rerun the protocol in our head using the fake CRS and we hope that the output collides with R. If we fail, we retry sampling a new fake CRS. Once we succeed, we output the fake σ and the randomness of the parties that led to the collision.

By the first corollary of Theorem 1.1, we know that, *on average* over R, the collision probability is $1/\mathsf{poly}(\lambda)$. So, for a *polynomial fraction* of all possible values R, the simulation succeeds after a polynomial number of tries. For the remaining fraction of the support of $\mathcal{D}$, our approach fails, meaning that the CRS and the randomness of the parties might leak too much information about the output.

In other words, the sampling protocol we described is secure only for a polynomial fraction of the support of $\mathcal{D}$. The good news is that it is possible to tell if the result of our non-interactive sampling protocol lies in the secure subset or not: the parties can locally run the simulator. If it succeeds with sufficiently high frequency, they can be sure their output is secure, otherwise, they need to discard it, generate a new σ and public coins and rerun the protocol. Since there is a polynomial fraction of the support of $\mathcal{D}$ that will not be discarded, the players need a polynomial number of attempts before succeeding. We also point out that the distribution of the outputs will be biased, but not significantly: if $\mathcal{D}$ describes the distribution of another protocol's CRS, it is still secure to use the outputs of our procedure as CRSs for such protocol.

3 Preliminaries on Entropy

In information theory, entropy is used to measure the unpredictability of random variables. After almost a century of research, several definitions have been formalised. In this section, we recall some of the important notions and the related properties. We start with Shannon's entropy [Sha48].

Definition 3.1 (Shannon's entropy). *Let $\mathbf{X}$ be a random variable having finite support. The Shannon's entropy of $\mathbf{X}$ is*

$$\mathsf{H}(\mathbf{X}) := -\sum_{x} \Pr[\mathbf{X} = x] \cdot \log\big(\Pr[\mathbf{X} = x]\big).$$

We recall also the notion of conditional Shannon's entropy, mutual information and interaction information.

Definition 3.2 (Conditional Shannon's entropy). *Let $\mathbf{X}$ and $\mathbf{Y}$ be random variables having finite support and let E be an event. The Shannon's entropy of $\mathbf{X}$ conditioned on E is*

$$\mathsf{H}(\mathbf{X}|E) := -\sum_{x} \Pr[\mathbf{X} = x|E] \cdot \log\big(\Pr[\mathbf{X} = x|E]\big).$$

The Shannon's entropy of $\mathbf{X}$ conditioned on $\mathbf{Y}$ is instead

$$\mathsf{H}(\mathbf{X}|\mathbf{Y}) := \sum_{y} \Pr[\mathbf{Y} = y] \cdot \mathsf{H}(\mathbf{X}|\mathbf{Y} = y).$$

Definition 3.3 (Mutual information). *Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ be random variables. The mutual information of $\mathbf{X}$ and $\mathbf{Y}$ conditioned on $\mathbf{Z}$ is*

$$\mathsf{I}(\mathbf{X};\mathbf{Y}|\mathbf{Z}) := \mathsf{H}(\mathbf{X}|\mathbf{Z}) - \mathsf{H}(\mathbf{X}|\mathbf{Y},\mathbf{Z}).$$

Definition 3.4 (Interaction information). *Let* $\mathbf{X}_1, \ldots, \mathbf{X}_n$ *and* $\mathbf{Z}$ *be random variables. The interaction information of* $\mathbf{X}_1, \ldots, \mathbf{X}_n$ *conditioned on* $\mathbf{Z}$ *is defined inductively by* $\mathsf{I}(\mathbf{X}_1|\mathbf{Z}) := \mathsf{H}(\mathbf{X}_1|\mathbf{Z})$ *and*

$$\mathsf{I}(\mathbf{X}_1; \ldots; \mathbf{X}_n|\mathbf{Z}) := \mathsf{I}(\mathbf{X}_1; \ldots; \mathbf{X}_{n-1}|\mathbf{Z}) - \mathsf{I}(\mathbf{X}_1; \ldots; \mathbf{X}_{n-1}|\mathbf{X}_n, \mathbf{Z}).$$

Shannon's entropy satisfies an important property called *the strong chain rule*. We recall it below.

Theorem 3.1 (Strong chain rule). *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be random variables with finite support. Then,*

$$\mathsf{H}(\mathbf{X}, \mathbf{Y}) = \mathsf{H}(\mathbf{Y}) + \mathsf{H}(\mathbf{X}|\mathbf{Y}).$$

Notice that $(\mathbf{X}, \mathbf{Y})$ is a random variable, so $\mathsf{H}(\mathbf{X}, \mathbf{Y})$ is defined as in Def. 3.1. We also recall the following properties of Shannon's entropy.

Lemma 3.1. *Let* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{Z}$ *be random variables with finite support. Then,*

1. *If* $\mathbf{X}$ *is uniform over a set of cardinality* m, $\mathsf{H}(\mathbf{X}) = \log m$.
2. *If* $\mathbf{X}$ *is independent of* $\mathbf{Y}$, *given* $\mathbf{Z}$, $\mathsf{H}(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) = \mathsf{H}(\mathbf{X}|\mathbf{Z})$ *and so* $\mathsf{I}(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) = 0$.
3. $\mathsf{H}(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) \leq \mathsf{H}(\mathbf{X}|\mathbf{Z})$ *and so* $\mathsf{I}(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) \geq 0$.
4. *If* f *is a deterministic function,* $\mathsf{H}(f(\mathbf{X})) \leq \mathsf{H}(\mathbf{X})$.

We now recall other definitions of entropy that are used to prove our results.

Definition 3.5 (Max entropy). *Let* X *be a random variable with finite support, let* E *be an event. We define the max entropy of* $\mathbf{X}$ *to be*

$$\mathsf{H}_0(\mathbf{X}) = \log|\mathsf{Supp}(\mathbf{X})|.$$

We define the max entropy of $\mathbf{X}$ *conditioned on* E *to be*

$$\mathsf{H}_0(\mathbf{X}|E) = \log|\mathsf{Supp}(\mathbf{X}|E)|.$$

Definition 3.6 (Min entropy). *Let* $\mathbf{X}$ *be a random variable with finite support, let* E *be an event. We define the min entropy of* $\mathbf{X}$ *to be*

$$\mathsf{H}_\infty(\mathbf{X}) = -\log\big(\max_x \Pr[\mathbf{X} = x]\big).$$

We define the min entropy of X *conditioned on* E *to be*

$$\mathsf{H}_\infty(\mathbf{X}|E) = -\log\big(\max_x \Pr[\mathbf{X} = x|E]\big).$$

Finally, we recall the definition of collision entropy.

Definition 3.7 (Collision entropy). *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be random variables with finite support, let* E *be an event. We define the collision entropy of* $\mathbf{X}$ *to be*

$$\mathsf{H}_2(\mathbf{X}) = -\log\Big(\sum_x \Pr[\mathbf{X} = x]^2\Big) = -\log\big(\Pr[\mathbf{X} = \mathbf{X}']\big),$$

where $\mathbf{X}'$ *is independent and identically distributed to* $\mathbf{X}$. *We define the collision entropy of* $\mathbf{X}$ *conditioned on* E *to be*

$$\mathsf{H}_2(\mathbf{X}|E) = -\log\Big(\sum_x \Pr[\mathbf{X} = x|E]^2\Big).$$

The average collision entropy of $\mathbf{X}$ *given* $\mathbf{Y}$ *is instead*

$$\widetilde{\mathsf{H}}_2(\mathbf{X}|\mathbf{Y}) = -\log\Big(\sum_{x,y} \Pr[\mathbf{Y} = y] \cdot \Pr[\mathbf{X} = x|\mathbf{Y} = y]^2\Big).$$

All the above definitions of entropy are not equivalent. For instance, Shannon's entropy can assume values that are significantly larger than min and collision entropy. The definitions are however related by the following well-known inequalities.

Theorem 3.2. *Let* $\mathbf{X}$ *be a random variable with finite support, let* E *be an event. We have that*

$$0 \leq \mathsf{H}_\infty(\mathbf{X}) \leq \mathsf{H}_2(\mathbf{X}) \leq \mathsf{H}(\mathbf{X}) \leq \mathsf{H}_0(\mathbf{X}),$$

$$0 \leq \mathsf{H}_\infty(\mathbf{X}|E) \leq \mathsf{H}_2(\mathbf{X}|E) \leq \mathsf{H}(\mathbf{X}|E) \leq \mathsf{H}_0(\mathbf{X}|E) \leq \mathsf{H}_0(\mathbf{X}).$$

Yao's Incompressibility Entropy. All the entropy notions we recalled above are great for measuring information theoretic properties, however, they all suffer from an important disadvantage, namely, they do not behave well under computational indistinguishability. Specifically, if $\mathbf{X} \sim_c \mathbf{X}'$, the entropy of $\mathbf{X}$ can be significantly different from the entropy of $\mathbf{X}'$, it does not matter which of the above definitions we consider.

We solve this issue by relying on a notion of computational entropy [Yao82, HLR07]. We recall the definition.

Definition 3.8 (Yao's entropy). *Let $(\mathbf{X}_\lambda)_{\lambda\in\mathbb{N}}$ be an ensemble of random variables. We say that the Yao entropy of $\mathbf{X}$ is smaller or equal to $k(\lambda)$, written $\mathsf{H}_{\mathsf{Yao}}(\mathbf{X}) \leq k(\lambda)$, if there exists a pair of polynomial sized deterministic circuits $(c_\lambda, d_\lambda)_{\lambda\in\mathbb{N}}$ such that*

$$\Pr[d_\lambda(c_\lambda(\mathbf{X})) = \mathbf{X}] \geq \frac{2^{\ell(\lambda)}}{2^{k(\lambda)}} - \mathsf{negl}(\lambda).$$

In the above formula, $\ell(\lambda)$ denotes the output size of c_λ. The circuit c_λ is called a compressor, whereas d_λ is called a decompressor.

In [HLR07], Hsiao, Lu and Reyzin generalised the definition to the conditional case. We recall it below.

Definition 3.9 (Conditional Yao's entropy). *Let $(\mathbf{X}_\lambda)_{\lambda\in\mathbb{N}}$ and $(\mathbf{Y}_\lambda)_{\lambda\in\mathbb{N}}$ be two ensembles of random variables. We say that the Yao entropy of X conditioned on Y is smaller or equal to $k(\lambda)$, written $\mathsf{H}_{\mathsf{Yao}}(\mathbf{X}|\mathbf{Y}) \leq k(\lambda)$, if there exists a pair of polynomial sized deterministic circuits $(c_\lambda, d_\lambda)_{\lambda\in\mathbb{N}}$ such that*

$$\Pr[d(c(\mathbf{X}, \mathbf{Y}), \mathbf{Y}) = \mathbf{X}] \geq \frac{2^{\ell(\lambda)}}{2^{k(\lambda)}} - \mathsf{negl}(\lambda).$$

In the above formula, $\ell(\lambda)$ denotes the output size of c_λ. The circuit c_λ is called a compressor, whereas d_λ is called a decompressor. If $\mathsf{H}_{\mathsf{Yao}}(\mathbf{X}|\mathbf{Y}) \leq k(\lambda)$ where $k(\lambda)$ is $O(\log\lambda)$, we will simply write that $\mathsf{H}_{\mathsf{Yao}}(\mathbf{X}|\mathbf{Y}) = O(\log\lambda)$.

Essentially, Yao's incompressibility entropy measures how much it is possible to compress, in polynomial time, samples from a distribution $\mathbf{X}$ given that the outcome of the possibly correlated random variable $\mathbf{Y}$ is known.

We observe that Yao's entropy can assume values that are significantly larger than Shannon's entropy. Examples of this kind are the outputs of PRGs. In some particular cases, however, also the opposite is true. For instance, there exist distributions $\mathbf{X}$ such that $\mathsf{H}_\infty(\mathbf{X}) = O(\log\lambda)$ but $\mathsf{H}(\mathbf{X}) = \omega(\log\lambda)$. For all such $\mathbf{X}$, we have $\mathsf{H}_{\mathsf{Yao}}(\mathbf{X}) = O(\log\lambda)$ (consider the compressor that outputs the empty string and the decompressor that outputs the most likely element).

The following well-known lemma formalises the fact that Yao's entropy preserves under computational indistinguishability.

Lemma 3.2. *Let $(\mathbf{X}_\lambda, \mathbf{Y}_\lambda)_{\lambda\in\mathbb{N}}$ and $(\mathbf{X}'_\lambda, \mathbf{Y}'_\lambda)_{\lambda\in\mathbb{N}}$ be two ensembles of random variables such that $(\mathbf{X}_\lambda, \mathbf{Y}_\lambda) \sim_c (\mathbf{X}'_\lambda, \mathbf{Y}'_\lambda)$. Then, $\mathsf{H}_{\mathsf{Yao}}(\mathbf{X}|\mathbf{Y}) \leq k(\lambda)$ if and only if $\mathsf{H}_{\mathsf{Yao}}(\mathbf{X}'|\mathbf{Y}') \leq k(\lambda)$.*

We highlight that Yao's entropy is not the only notion of computational entropy [Rey11, HILL99, HLR07]. Among all the studied notions, it is however the one assuming highest values [HLR07]. We decided to use Yao's entropy exactly for this reason, making the results presented in this paper as strong as possible.

4 Impossibility of Distributed Samplers without Random Oracle

We now present and prove our main theorem, namely that in a strong semi-malicious distributed sampler $\mathsf{H}(R|\sigma) = O(\log\lambda)$. The idea was sketched in the technical overview (see Section 2.1).

We start by describing what an n-party distributed sampler with black-box simulation is.

Definition 4.1 (Syntax of distributed samplers). *Let $\mathcal{D}(1\!\!1^\lambda)$ be an efficiently samplable distribution. An n-party distributed sampler for $\mathcal{D}(1\!\!1^\lambda)$ consists of a triple of PPT algorithms $(\mathsf{CRS}, \mathsf{Gen}, \mathsf{Sample})$ with the following syntax:*

– CRS *is randomised, it takes as input the security parameter* $\mathbb{1}^\lambda$ *and outputs a CRS* σ*.*
– Gen *is randomised and takes as input the security parameter* $\mathbb{1}^\lambda$*, a CRS* σ *and the index of the party* j *running the procedure. The output is a distributed sampler message* U_j *and secret information* r_j*.*
– Sample *is deterministic and takes as input a CRS* σ*,* n *distributed sampler messages* $U_1, \dots, U_n$*, the index of a party* j *and the relative secret information* r_j*. The output is a sample* R *from the distribution* $\mathcal{D}(\mathbb{1}^\lambda)$*.*

We say that the distributed sampler has public output if Gen *always outputs* $r_j = \bot$*.*

Definition 4.2 (t-out-of-n distributed sampler with black-box simulation). *Let* $\mathcal{C}$ *be an adversarial class and let* $t \leq n$*. We say that an* n*-party distributed sampler* $(\mathsf{CRS}, \mathsf{Gen}, \mathsf{Sample})$ *for the distribution* $\mathcal{D}(\mathbb{1}^\lambda)$ *is* t*-out-of-*n *secure against* $\mathcal{C}$ *with black-box simulation if there exists a uniform PPT simulator* Sim *such that, for every polynomial function* $q(\lambda)$*, uniform adversary* $\mathcal{A} \in \mathcal{C}$ *requiring* $q(\lambda)$ *random coins, auxiliary information* ψ *and every subset* $H \subseteq [n]$ *such that* $|H| \geq n - t$*, the following distributions are computationally indistinguishable*

$$\left\{ \begin{array}{c|l} & \rho \xleftarrow{\$} \{0,1\}^{q(\lambda)} \\ (R_j)_{j\in H} & \sigma \xleftarrow{\$} \mathsf{CRS}(\mathbb{1}^\lambda) \\ \sigma, U_H, \rho & \forall j \in H : (U_j, r_j) \xleftarrow{\$} \mathsf{Gen}(\mathbb{1}^\lambda, \sigma, j) \\ & U_C \leftarrow \mathcal{A}(\mathbb{1}^\lambda, \sigma, U_H, \psi; \rho) \\ & \forall j \in H : R_j \leftarrow \mathsf{Sample}(\sigma, U_1, \dots, U_n, j, r_j) \end{array} \right\}$$

$$\left\{ \begin{array}{c|l} (R_j)_{j\in H} & (\widehat{\mathsf{id}}, U_H, \sigma, \rho) \xleftarrow{\$} \mathsf{Sim}^{\mathcal{A}, \mathcal{F}_{\mathcal{D}}}(\mathbb{1}^\lambda, H, \psi) \\ \sigma, U_H, \rho & \forall j \in H : R_j \leftarrow R_{\widehat{\mathsf{id}}} \end{array} \right\}$$

Notice that Sim *is given oracle access to* $\mathcal{A}$ *and* $\mathcal{F}_{\mathcal{D}}$ *(see Fig. 1). This means that it can query* $\mathcal{A}$*, multiple times, on any input* $(\mathbb{1}^{\lambda'}, \sigma', U'_H, \rho')$ *and see the corresponding response. Furthermore,* Sim *can ask* $\mathcal{F}_{\mathcal{D}}$ *for samples. We use* $R_{\widehat{\mathsf{id}}}$ *to denote the sample produced by* $\mathcal{F}_{\mathcal{D}}$ *associated with the label* $\widehat{\mathsf{id}}$*. If such sample does not exist, we set* $R_{\widehat{\mathsf{id}}} := \bot$*.*

Remark 4.1. There is a technicality we need to pay attention to: what if the simulator runs in time $p(\lambda) < q(\lambda)$? It would be impossible for the simulator to query the adversary even once, let alone outputting the randomness ρ at the end of its execution. This fact was already observed by Bellare, Micali and Ostrovsky in [BMO90]. To solve the issue, they proposed the following modification: suppose that the simulator is equipped with two random tapes, one for itself and one for the adversary (which is provided with $q(\lambda)$ random coins). The simulator is allowed to read and modify the random tape of the adversary as long as its total running time does not exceed $p(\lambda)$ (in particular, the simulator can touch at most $p(\lambda)$ bits on the adversary's tape). Each time the simulator queries $\mathcal{A}$, the adversary is run using the randomness present on its random tape at that particular moment. Moreover, the randomness ρ output by the simulator will be the one present on the adversary's random tape at the end of the simulator's execution.

There are other ways this problem could be addressed, for instance, by setting an upper-bound on the size of the random tape of the adversary. Even if it is unclear whether all these definitions are equivalent, we believe that our impossibility can be extended (perhaps, with some non-trivial argument) also to these models. The reason why we chose this particular definition of black-box simulation is that it is the one that allows us to prove, in the most direct way, an intuition common to all black-box simulation settings: if the simulator rewinds the adversary on some new input, the adversary generates its answer using independent looking randomness.

Definition 4.3 (Strong semi-malicious security). *We say that a distributed sampler is strongly semi-maliciously secure if it is secure against the class of uniform adversaries that, after receiving* σ*,* U_H *and auxiliary information* ψ*, output* $(U_j)_{j\in C}$ *where*

$$\forall j \in C : U_C \leftarrow \mathsf{Gen}(\mathbb{1}^\lambda, \sigma, j; \rho_j), \qquad (\rho_j)_{j\in C} \xleftarrow{\$} \mathcal{M}(\mathbb{1}^\lambda, \sigma, U_H, \psi)$$

and $\mathcal{M}$ *is a PPT algorithm.*

Theorem 4.1. *Let $\mathcal{D}(\mathbb{1}^\lambda)$ be an efficient distribution. In a strongly semi-maliciously secure, t-out-of-n distributed sampler for $\mathcal{D}(\mathbb{1}^\lambda)$ with black-box simulation, we have that $\mathsf{H}(R_i \,|\, \sigma) = \frac{n}{t} \cdot O(\log \lambda)$ for any $i \in [n]$, where R_i denotes the output of party P_i.*

Before proving the above theorem, we state and prove the following lemma about information theory.

Lemma 4.1. *Let $\mathbf{X}_1, \dots, \mathbf{X}_n, \mathbf{Y}$ be random variables such that $\mathbf{X}_1, \dots, \mathbf{X}_n$ are all independent conditioned on $\mathbf{Y}$. Let f be a deterministic function of $\mathbf{X}_1, \dots, \mathbf{X}_n, \mathbf{Y}$. Then, if we denote $f(\mathbf{X}_1, \dots, \mathbf{X}_n, \mathbf{Y})$ by $\mathbf{R}$, we have that, for every $t \in [n]$, there exists $S \subseteq [n]$ such that $|S| = t$ and*

$$\mathsf{H}(\mathbf{R}|\mathbf{X}_S, \mathbf{Y}) \geq \frac{n-t}{n} \cdot \mathsf{H}(\mathbf{R}|\mathbf{Y})$$

Proof. Fix a subset $S' \subseteq [n]$ such that $|S'| = t-1$. Consider also disjoint subsets $S_0, S_1 \subseteq [n]$. Due to the independence between $\mathbf{X}_1, \dots, \mathbf{X}_n$ conditioned on $\mathbf{Y}$, we have that, for any $i \in S_1$,

$$\begin{aligned}
&\mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1} | \mathbf{X}_{S_0}, \mathbf{Y}) \\
&\overset{3.4}{=} \mathsf{I}(\mathbf{R}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y}) + \mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1 \setminus \{i\}} | \mathbf{X}_{S_0 \cup \{i\}}, \mathbf{Y}) \\
&\overset{3.4}{=} \mathsf{I}(\mathbf{R}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y}) + \mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1 \setminus \{i\}} | \mathbf{X}_{S_0}, \mathbf{Y}) - \mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1 \setminus \{i\}}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y}) \\
&\overset{3.4}{=} \mathsf{I}(\mathbf{R}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y}) + \mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1 \setminus \{i\}} | \mathbf{X}_{S_0}, \mathbf{Y}) - \Big(\mathsf{I}(\mathbf{X}_{S_1 \setminus \{i\}}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y}) - \mathsf{I}(\mathbf{X}_{S_1 \setminus \{i\}}; \mathbf{X}_i | \mathbf{R}, \mathbf{X}_{S_0}, \mathbf{Y})\Big) \\
&\overset{3.1.2}{=} \mathsf{I}(\mathbf{R}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y}) + \mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1 \setminus \{i\}} | \mathbf{X}_{S_0}, \mathbf{Y}) + \mathsf{I}(\mathbf{X}_{S_1 \setminus \{i\}}; \mathbf{X}_i | \mathbf{R}, \mathbf{X}_{S_0}, \mathbf{Y}) \\
&\overset{3.1.3}{\geq} \mathsf{I}(\mathbf{R}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y}) + \mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1 \setminus \{i\}} | \mathbf{X}_{S_0}, \mathbf{Y}).
\end{aligned}$$

By recursively applying the bound above to the term $\mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1 \setminus \{i\}} | \mathbf{X}_{S_0}, \mathbf{Y})$, we conclude that

$$\mathsf{I}(\mathbf{R}; \mathbf{X}_{S_1} | \mathbf{X}_{S_0}, \mathbf{Y}) \geq \sum_{i \in S_1} \mathsf{I}(\mathbf{R}; \mathbf{X}_i | \mathbf{X}_{S_0}, \mathbf{Y})$$

and so

$$\begin{aligned}
\mathsf{H}(\mathbf{R}|\mathbf{X}_{S'}, \mathbf{Y}) &\overset{3.1.2}{=} \mathsf{I}(\mathbf{R}; \mathbf{X}_{[n] \setminus S'} | \mathbf{X}_{S'}, \mathbf{Y}) \\
&\geq \sum_{i \notin S'} \mathsf{I}(\mathbf{R}; \mathbf{X}_i | \mathbf{X}_{S'}, \mathbf{Y}) \\
&\overset{3.3}{=} (n-t+1) \cdot \mathsf{H}(\mathbf{R}|\mathbf{X}_{S'}, \mathbf{Y}) - \sum_{i \notin S'} \mathsf{H}(\mathbf{R}|\mathbf{X}_{S' \cup \{i\}}, \mathbf{Y}).
\end{aligned}$$

This proves that $(n-t) \cdot \mathsf{H}(\mathbf{R}|\mathbf{X}_{S'}, \mathbf{Y}) \leq \sum_{i \notin S'} \mathsf{H}(\mathbf{R}|\mathbf{X}_{S' \cup \{i\}}, \mathbf{Y})$ and therefore

$$\begin{aligned}
\sum_{|S'|=t-1} (n-t) \cdot \mathsf{H}(\mathbf{R}|\mathbf{X}_{S'}, \mathbf{Y}) &\leq \sum_{|S'|=t-1} \sum_{i \notin S'} \mathsf{H}(\mathbf{R}|\mathbf{X}_{S' \cup \{i\}}, \mathbf{Y}) \\
&= t \cdot \sum_{|S|=t} \mathsf{H}(\mathbf{R}|\mathbf{X}_S, \mathbf{Y}).
\end{aligned}$$

Notice that the last equality follows from the fact that, given S such that $|S| = t$, there are t ways to choose S' and $i \notin S'$ such that $S = S' \cup \{i\}$. In other words, in the sum on the left, $\mathsf{H}(\mathbf{R}|\mathbf{X}_S, \mathbf{Y})$ gets added t times. Now, with an inductive argument over t, we obtain that

$$\begin{aligned}
\sum_{|S|=t} \mathsf{H}(\mathbf{R}|\mathbf{X}_S, \mathbf{Y}) &\geq \frac{n-t}{t} \cdot \sum_{|S|=t-1} \mathsf{H}(\mathbf{R}|\mathbf{X}_S, \mathbf{Y}) \\
&\vdots \\
&\geq \frac{n-t}{t} \cdot \frac{n-t+1}{t-1} \cdots \frac{n-1}{1} \cdot \mathsf{H}(\mathbf{R}|\mathbf{Y}).
\end{aligned}$$

Finally, by an averaging argument, we conclude that there exists S such that $|S| = t$ and

$$\mathsf{H}(\mathbf{R}|\mathbf{X}_S, \mathbf{Y}) \geq \frac{\binom{n-1}{t}}{\binom{n}{t}} \cdot \mathsf{H}(\mathbf{R}|\mathbf{Y}) = \frac{n-t}{n} \cdot \mathsf{H}(\mathbf{R}|\mathbf{Y}).$$

□

We are finally ready to prove Theorem 4.1.

Proof. Throughout the proof, we view $\boldsymbol{\sigma}$ and $\mathbf{U}_H$ as random variables, i.e., functions of the randomness of the various algorithms involved in the considered settings: real world, ideal world and a hybrid world, we are about to introduce.

Since the simulator runs in polynomial time, there exists a polynomial upper bound $q(\lambda)$ on the number of samples the simulator queries to the functionality. Let $\mathbf{Q}$ be the random variable describing the set containing the responses to these queries. There exists also a polynomial upper bound on the number of times the simulator queries the adversary, let this be $w(\lambda)$. Finally, there exists a polynomial upper bound $\ell(\lambda)$ on the number of bits on the adversary's random tape that are read or modified by the simulator. Without loss of generality, we can make three assumptions regarding the simulator. The first one is that it issues all its queries to the functionality at the beginning of its execution. The second one is that it never outputs a tuple $(\widehat{\mathsf{id}}, U_H, \sigma, \rho)$ without first having queried $\mathcal{A}$ on (U_H, σ, ρ). The last one is that the simulator never queries the adversary twice on the same value.

We consider the uniform adversary that follows the protocol but generates the randomness it feeds into Gen using a $w(\lambda)$-wise independent hash function applied on $(\mathbf{U}_H, \boldsymbol{\sigma})$. The hash key for such hash function is obtained by ignoring the first $\ell(\lambda)$ bits of the adversary's randomness. Notice that, from the simulator's perspective, the hash key is random, because it cannot read nor modify past the $\ell(\lambda)$-th element in the string.

We therefore consider a hybrid world obtained by modifying the ideal world: the queries of the simulator to the adversary will be now generated without using the w-wise independent hash function, but just using uniform randomness provided on a new tape not accessible to the simulator. More specifically, the queries of the simulator will now be generated by a stateful machine making sure that identical queries are answered in the same way, and new queries are answered using fresh randomness. From the perspective of the simulator, this hybrid is identical to the ideal world. Indeed, previously, the view of the simulator was independent of the w-wise independent hash key used by the adversary (the simulator could not even read such key). So, given that the adversary was queried at most $w(\lambda)$ times, all responses were generated with independent, uniform randomness. In this hybrid world, our probability space $\varXi$ will be the union of three probability subspaces: the one defined by the randomness of the simulator, the one defined by the randomness of the functionality and the one defined randomness of the new hybrid adversary.

Let $(\mathbf{U}_H^i, \boldsymbol{\sigma}^i, \boldsymbol{\rho}^i)$ be random variables describing the i-th query of the simulator to the adversary $\mathcal{A}$[10]. Let $\mathbf{U}_C^i$ be the random variable corresponding to the answer. Notice that, since our adversary generates the messages of the corrupted parties following the protocol, we can also define the random variables $(\mathbf{r}_j^i)_{j \in C}$ representing the secret information obtained by the corrupted parties during the generation of $\mathbf{U}_C^i$. Since there exists a unique $\boldsymbol{\mu} \in [w]$ such that $(\boldsymbol{\sigma}, \mathbf{U}_H) = (\boldsymbol{\sigma}^{\boldsymbol{\mu}}, \mathbf{U}_H^{\boldsymbol{\mu}})$, we define $\mathbf{U}_C$ as the random variable $\mathbf{U}_C^{\boldsymbol{\mu}}$. Similarly, for every $j \in C$, we define the random variable $\mathbf{r}_j$ as $\mathbf{r}_j^{\boldsymbol{\mu}}$.

Assume that Gen requires $L(\lambda)$ bits of randomness. Take $k \in \mathbb{N}$ such that $t \cdot L \leq \lambda^k$. Consider a constant $T \in \mathbb{N}$. Let ι be the index of a fixed corrupted party, we define the random variable $\mathbf{R}_\iota$ as $\mathsf{Sample}(\boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{U}_C, \iota, \mathbf{r}_\iota)$. Independently of the world we are working on, we consider an auxiliary probability space $\varXi_{\mathsf{aux}}$ providing uniform randomness (independent of the one used by the various actors in the protocol and in the simulation). We denote the probability measure over such space by $\Pr_{\mathsf{aux}}$. For any values $\widehat{R}, \widehat{U}_H, \widehat{\sigma}$, we define the following quantities:

$$\begin{aligned} p(\widehat{R}, \widehat{U}_H, \widehat{\sigma}) &:= \Pr_{\mathsf{aux}}\left[\mathsf{Sample}(\widehat{\sigma}, \widehat{U}_H, \mathbf{U}_C, \iota, \mathbf{r}_\iota) = \widehat{R} \middle| \forall j \in C: \quad (\mathbf{U}_j, \mathbf{r}_j) \xleftarrow{\$} \mathsf{Gen}(\mathbb{1}^\lambda, \widehat{\sigma}, j)\right] \\ p_T(\lambda) &:= \Pr\left[p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-T}\right] \end{aligned}$$

Notice that $p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma})$ is a random variable over the probability space $\varXi$.

[10] Notice that we are not fixing the values of these variables, we are just giving a name to the functions that map the randomness in $\varXi$ to the i-th query of the simulator to the adversary.

Claim 4.1.1. *For any $T \in \mathbb{N}$, in the real world, we have*

$$\mathsf{H}(\mathbf{R}_\iota | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \leq O(\lambda^k) \cdot p_T(\lambda) + T \cdot \log \lambda$$

Proof of the claim. Let $\mathbf{E}$ be the random variable that is equal to 1 when $p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) \geq \lambda^{-T}$, 0 otherwise. We observe that

$$\begin{aligned}
&\mathsf{H}(\mathbf{R}_\iota | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.1.4}{=} \mathsf{H}(\mathbf{R}_\iota, \mathbf{E} | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.1}{=} \mathsf{H}(\mathbf{R}_\iota | \mathbf{E}, \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + \mathsf{H}(\mathbf{E} | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.2}{\leq} \mathsf{H}(\mathbf{R}_\iota | \mathbf{E}, \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + \mathsf{H}_0(\mathbf{E} | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.5}{\leq} \mathsf{H}(\mathbf{R}_\iota | \mathbf{E}, \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + 1 \\
&\overset{3.2}{=} \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E} = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E} = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} + \mathsf{H}\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E} = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E} = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} \right) + 1 \\
&\leq \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E} = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) + \mathsf{H}\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E} = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \left[\mathbf{E} = 0 \middle| \begin{matrix} \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right] \right) \cdot \Pr \begin{bmatrix} \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} + 1 \\
&\overset{3.2}{\leq} \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}_0\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E} = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) + \mathsf{H}_0\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E} = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \left[\mathbf{E} = 0 \middle| \begin{matrix} \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right] \right) \cdot \Pr \begin{bmatrix} \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} + 1 \\
&\overset{3.5}{=} \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(T \cdot \log \lambda + O(\lambda^k) \cdot \Pr \left[\mathbf{E} = 0 \middle| \begin{matrix} \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right] \right) \cdot \Pr \begin{bmatrix} \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} + 1 \\
&= O(\lambda^k) \cdot p_T(\lambda) + T \cdot \log \lambda.
\end{aligned}$$

The second last inequality follows from the fact that there could be at most λ^T elements $\mathbf{R}_\iota$ such that $p(\mathbf{R}_\iota, \widehat{U}_H, \widehat{\sigma}) \geq \lambda^{-T}$. Moreover, the total number of possible outputs, once conditioned on $\boldsymbol{\sigma} = \widehat{\sigma}$, is bounded from above by the total amount of randomness used by the parties. ∎

Claim 4.1.2. *In the hybrid world, we have*

$$\Pr \left[p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-T} \middle| (\widehat{\mathsf{id}}, \mathbf{U}_H, \boldsymbol{\sigma}, \boldsymbol{\rho}) \xleftarrow{\$} \mathsf{Sim}^{\mathcal{A}, \mathcal{F}_{\mathcal{D}}}(\mathbb{1}^\lambda) \right] \leq w(\lambda) \cdot q(\lambda) \cdot \lambda^{-T} + \mathsf{negl}(\lambda).$$

Proof of the claim. Let E' be the event in which $\mathbf{R}_{\widehat{\mathsf{id}}} = \mathsf{Sample}(\boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{U}_C, \iota, \mathbf{r}_\iota)$. Observe that, even if P_ι is corrupted, $\mathbf{r}_\iota$ is still well-defined, given that our adversary $\mathcal{A}$ follows the protocol. We notice that $\Pr[E'] = 1 - \mathsf{negl}(\lambda)$. Indeed, consider the protocol execution when no party is corrupted: in the ideal world, the functionality outputs the same sample R to all parties, therefore, with overwhelming probability, all parties output the same sample R even in the real world. Now, let's go back to our setting. Even if some of our parties are corrupted, they behave as if they were honest. So, in the real world, with overwhelming probability, the outputs of the honest parties will all be the same and will coincide with $\mathsf{Sample}(\boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{U}_C, j, \mathbf{r}_j)$ for every corrupted party P_j. Notice that, as for P_ι, even if P_j is corrupted, $\mathbf{r}_j$ is still well-defined, given that we only consider adversaries that follow the protocol. We argue that, even in the ideal world (where some parties are corrupted), with overwhelming probability, the honest parties all output the same sample $\mathbf{R}$ and this will coincide with $\mathsf{Sample}(\boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{U}_C, j, \mathbf{r}_j)$ for every corrupted party P_j. If that was not the case, we would have an easy way to distinguish the real world from the ideal world. We conclude by observing that, in the ideal world, the output of the honest parties is $\mathbf{R}_{\widehat{\mathsf{id}}}$. This ensures that $\Pr[E'] = 1 - \mathsf{negl}(\lambda)$. Finally, notice that the distribution of $(\boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{U}_C, \mathbf{r}_C, \mathbf{R}_{\widehat{\mathsf{id}}})$ in the hybrid world is the same as in the ideal world. This proves that, even in such setting, $\Pr[E'] = 1 - \mathsf{negl}(\lambda)$.

Next, fix an $i \in [w]$. We define the random variable $\mathbf{R}^i_\iota = \mathsf{Sample}(\boldsymbol{\sigma}^i, \mathbf{U}^i_H, \mathbf{U}^i_C, \iota, \mathbf{r}^i_\iota)$. Consider any values $\widehat{Q}, \widehat{R}, \widehat{U}_H, \widehat{\sigma}$ (not random variables) such that $p(\widehat{R}, \widehat{U}_H, \widehat{\sigma}) < \lambda^{-T}$. We observe that, trivially,

$$\Pr\left[\mathsf{Sample}(\boldsymbol{\sigma}^i, \mathbf{U}^i_H, \mathbf{U}^i_C, \iota, \mathbf{r}^i_\iota) = \widehat{R} \,\middle|\, \begin{matrix} \mathbf{Q} = \widehat{Q} \\ \boldsymbol{\sigma}^i = \widehat{\sigma} \\ \mathbf{U}^i_H = \widehat{U}_H \end{matrix}\right] < \lambda^{-T}.$$

This is because, in the hybrid world, the adversary generates $\mathbf{U}^i_C$ using uniform randomness, independent of $\mathbf{Q}, \mathbf{U}^i_H$ and $\boldsymbol{\sigma}^i$. We notice that

$$\begin{aligned}
&\Pr\left[\begin{matrix} p(\mathbf{R}^i_\iota, \mathbf{U}^i_H, \boldsymbol{\sigma}^i) < \lambda^{-T} \\ \mathbf{R}^i_\iota \in \mathbf{Q} \end{matrix}\right] \\
&= \sum_{\widehat{Q}, \widehat{U}_H, \widehat{\sigma}} \Pr\left[\begin{matrix} p(\mathbf{R}^i_\iota, \mathbf{U}^i_H, \boldsymbol{\sigma}^i) < \lambda^{-T} \\ \mathbf{R}^i_\iota \in \mathbf{Q} \end{matrix} \,\middle|\, \begin{matrix} \mathbf{Q} = \widehat{Q} \\ \mathbf{U}^i_H = \widehat{U}_H \\ \boldsymbol{\sigma}^i = \widehat{\sigma} \end{matrix}\right] \cdot \Pr\left[\begin{matrix} \mathbf{Q} = \widehat{Q} \\ \mathbf{U}^i_H = \widehat{U}_H \\ \boldsymbol{\sigma}^i = \widehat{\sigma} \end{matrix}\right] \\
&\leq \sum_{\widehat{Q}, \widehat{U}_H, \widehat{\sigma}} \quad \sum_{\substack{\widehat{R} \in \widehat{Q} \\ p(\widehat{R}, \widehat{U}_H, \widehat{\sigma}) < \lambda^{-T}}} \Pr\left[\mathsf{Sample}(\boldsymbol{\sigma}, \mathbf{U}^i_H, \mathbf{U}^i_C, \iota, \mathbf{r}^i_\iota) = \widehat{R} \,\middle|\, \begin{matrix} \mathbf{Q} = \widehat{Q} \\ \mathbf{U}^i_H = \widehat{U}_H \\ \boldsymbol{\sigma}^i = \widehat{\sigma} \end{matrix}\right] \cdot \Pr\left[\begin{matrix} \mathbf{Q} = \widehat{Q} \\ \mathbf{U}^i_H = \widehat{U}_H \\ \boldsymbol{\sigma}^i = \widehat{\sigma} \end{matrix}\right] \\
&< \sum_{\widehat{Q}, \widehat{U}_H, \widehat{\sigma}} \quad \sum_{\substack{\widehat{R} \in \widehat{Q} \\ p(\widehat{R}, \widehat{U}_H, \widehat{\sigma}) < \lambda^{-T}}} \lambda^{-T} \cdot \Pr[\mathbf{Q} = \widehat{Q}, \mathbf{U}^i_H = \widehat{U}_H, \boldsymbol{\sigma}^i = \widehat{\sigma}] \\
&\leq \sum_{\widehat{Q}, \widehat{U}_H, \widehat{\sigma}} \sum_{\widehat{R} \in \widehat{Q}} \lambda^{-T} \cdot \Pr[\mathbf{Q} = \widehat{Q}, \mathbf{U}^i_H = \widehat{U}_H, \boldsymbol{\sigma}^i = \widehat{\sigma}] \\
&\leq \sum_{\widehat{Q}, \widehat{U}_H, \widehat{\sigma}} q(\lambda) \cdot \lambda^{-T} \cdot \Pr[\mathbf{Q} = \widehat{Q}, \mathbf{U}^i_H = \widehat{U}_i, \boldsymbol{\sigma}^i = \widehat{\sigma}] \\
&= q(\lambda) \cdot \lambda^{-T}.
\end{aligned}$$

By union bounding over all $i \in [w]$, we conclude that

$$\Pr\left[\exists i \in [w] \text{ s.t. } \begin{matrix} p(\mathbf{R}^i_\iota, \mathbf{U}^i_H, \boldsymbol{\sigma}^i) < \lambda^{-T} \\ \mathbf{R}^i_\iota \in \mathbf{Q} \end{matrix}\right] \leq w(\lambda) \cdot q(\lambda) \cdot \lambda^{-T}.$$

Finally, we terminate the proof by observing that

$$\begin{aligned}
&\Pr\left[p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-T} \,\middle|\, (\widehat{\mathsf{id}}, \mathbf{U}_H, \boldsymbol{\sigma}, \boldsymbol{\rho}) \xleftarrow{\$} \mathsf{Sim}^{\mathcal{A}, \mathcal{F}_{\mathcal{D}}}(\mathbb{1}^\lambda)\right] \\
&\leq \Pr\left[p(\mathbf{R}_\iota, \mathbf{U}_H, \sigma) < \lambda^{-T} \,\middle|\, \begin{matrix} (\widehat{\mathsf{id}}, \mathbf{U}_H, \boldsymbol{\sigma}, \boldsymbol{\rho}) \xleftarrow{\$} \mathsf{Sim}^{\mathcal{A}, \mathcal{F}_{\mathcal{D}}}(\mathbb{1}^\lambda) \\ E' \end{matrix}\right] + \Pr[\neg E'] \\
&\leq \Pr\left[\exists i \in [w] \text{ s.t. } \begin{matrix} p(\mathbf{R}^i_\iota, \mathbf{U}^i_H, \boldsymbol{\sigma}^i) < \lambda^{-T} \\ \mathbf{R}^i_\iota \in \mathbf{Q} \end{matrix} \,\middle|\, E'\right] + \Pr[\neg E'] \\
&\leq \Pr\left[\exists i \in [w] \text{ s.t. } \begin{matrix} p(\mathbf{R}^i_\iota, \mathbf{U}^i_H, \boldsymbol{\sigma}^i) < \lambda^{-T} \\ \mathbf{R}^i_\iota \in \mathbf{Q} \end{matrix}\right] + 2 \cdot \Pr[\neg E'] \\
&\leq w(\lambda) \cdot q(\lambda) \cdot \lambda^{-T} + \mathsf{negl}(\lambda).
\end{aligned}$$

■

Claim 4.1.3. *In the real world,* $\mathsf{H}(\mathbf{R}_\iota \,|\, \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) = O(\log\lambda)$.

Proof of the claim. Suppose the claim is false. We pick $T > 1$ so that $w(\lambda) \cdot q(\lambda) \cdot \lambda^{-T} = O(\lambda^{-2k})$. We consider the distinguisher D that, given σ, U_H and the output R_ι, performs the following attacks

```
1. $\alpha \leftarrow 0$
2. for $i = 1, \ldots, \lambda^{4T}$:
   (a) $\forall j \in C: \quad (U'_j, r'_j) \xleftarrow{\$} \mathsf{Gen}(\mathbb{1}^\lambda, \sigma, j)$
   (b) $R'_\iota \leftarrow \mathsf{Sample}(\sigma, U_H, U'_C, \iota, r'_\iota)$
   (c) if $R'_\iota = R_\iota$, $\alpha \leftarrow \alpha + 1$
3. if $\alpha \geq \lambda^{3T} - \lambda^{2T+1}$, output 0
4. otherwise output 1.
```

Observe that the randomness of this distinguisher defines a new probability space Ξ_{D}. Let $\Pr_{\mathsf{D}}$ be the corresponding probability measure. Let $\Pr_{\mathsf{tot}}$ be the probability measure over the union of Ξ and Ξ_{D}.

By the Chernoff bound, we know that

$$\Pr_{\mathsf{D}}\left[|\alpha - p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) \cdot \lambda^{4T}| \leq \lambda^{2T+1}\right] \geq 1 - \mathsf{negl}(\lambda).$$

So, if $p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) \geq \lambda^{-T}$, with overwhelming probability over the randomness of the distinguisher, $\alpha \geq \lambda^{3T} - \lambda^{2T+1}$ and the algorithm outputs 0. In other words,

$$\Pr_{\mathsf{tot}}\left[\mathsf{D}(\mathbb{1}^\lambda, \boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{R}_\iota) = 1 \middle| p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) \geq \lambda^{-T}\right] \leq \mathsf{negl}(\lambda). \tag{1}$$

Now, we can upper-bound the total probability (over the randomness of the distinguisher, the simulator, the adversary and the functionality) that the distinguisher outputs 1 in the hybrid world by λ^{-2k}. Indeed, by putting together (1) and Claim 4.1.2, we have

$$\begin{aligned}
&\Pr_{\mathsf{tot}}\left[\mathsf{D}(\mathbb{1}^\lambda, \boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{R}_\iota) = 1\right] \\
&\leq \Pr_{\mathsf{tot}}\left[\mathsf{D}(\mathbb{1}^\lambda, \boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{R}_\iota) = 1 \middle| p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) \geq \lambda^{-T}\right] + \Pr\left[p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-T}\right] \\
&\leq \lambda^{-2k} + \mathsf{negl}(\lambda).
\end{aligned}$$

On the other hand, the total probability that the distinguisher outputs 1 in the real world is at least $\Theta(\lambda^{-k})$. Indeed, from Claim 4.1.1, we immediately deduce that, for every $T \in \mathbb{N}$,

$$p_T(\lambda) = \omega(\lambda^{-k}).$$

Moreover, we know that

$$\Pr\left[p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-2T}\right] = p_{2T}(\lambda) = \omega(\lambda^{-k}).$$

Notice that, by the Chernoff bound, when $p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-2T}$, with overwhelming probability over the randomness of the distinguisher, $\alpha < \lambda^{2T} + \lambda^{2T+1}$, which is asymptotically smaller than $\lambda^{3T} - \lambda^{2T+1}$. When this occurs, the distinguisher always outputs 1. In other words,

$$\Pr_{\mathsf{tot}}\left[\mathsf{D}(\mathbb{1}^\lambda, \boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{R}_\iota) = 1 \middle| p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-2T}\right] \geq 1 - \mathsf{negl}(\lambda).$$

To conclude,

$$\begin{aligned}
\Pr_{\mathsf{tot}}\left[\mathsf{D}(\mathbb{1}^\lambda, \boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{R}_\iota)\right] \geq &\Pr_{\mathsf{tot}}\left[\mathsf{D}(\mathbb{1}^\lambda, \boldsymbol{\sigma}, \mathbf{U}_H, \mathbf{R}_\iota) = 1 \middle| p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-2T}\right] \cdot \Pr\left[p(\mathbf{R}_\iota, \mathbf{U}_H, \boldsymbol{\sigma}) < \lambda^{-2T}\right] \\
\geq &\omega(\lambda^{-k}).
\end{aligned}$$

We have just proven that the advantage of our distinguisher is non-negligible! This is a contradiction. ■

Claim 4.1.4. *In the real world,* $\mathsf{H}(\mathbf{R}_i \,|\, \boldsymbol{\sigma}) = O(\log\lambda)$.

Proof of the claim. We observe that $\mathbf{R}_i$ is a deterministic function of $\boldsymbol{\sigma}, (\mathbf{U}_1, \mathbf{r}_1), \ldots, (\mathbf{U}_n, \mathbf{r}_n)$. Furthermore, $(\mathbf{U}_1, \mathbf{r}_1), \ldots, (\mathbf{U}_n, \mathbf{r}_n)$ are all independent conditioned on the CRS $\boldsymbol{\sigma}$. By applying Lemma 4.1, we obtain that there exists a subset $H \subseteq [n]$ such that $|H| = n - t$ and

$$\mathsf{H}(\mathbf{R}_i | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \geq \frac{t}{n} \cdot \mathsf{H}(\mathbf{R}_i | \boldsymbol{\sigma}).$$

We also observe that, with overwhelming probability, we have $R_1 = \cdots = R_n$. This is because, when all the parties are honest, in the ideal world execution, the simulator outputs the same sample to all the parties. If the same did not happen with overwhelming probability even in the real world, we would obtain a successful distinguisher.

Now, pick $\iota \notin H$, and let $\mathbf{E}'$ be the random variable that is equal to 1 when $\mathbf{R}_1 = \cdots = \mathbf{R}_n$, and 0 in all other cases. We observe that

$$\begin{aligned}
&\mathsf{H}(\mathbf{R}_i | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.1}{\leq} \mathsf{H}(\mathbf{R}_i, \mathbf{E}' | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.1}{=} \mathsf{H}(\mathbf{R}_i | \mathbf{E}', \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + \mathsf{H}(\mathbf{E}' | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.2}{\leq} \mathsf{H}(\mathbf{R}_i | \mathbf{E}', \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + \mathsf{H}_0(\mathbf{E}' | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) \\
&\overset{3.5}{\leq} \mathsf{H}(\mathbf{R}_i | \mathbf{E}', \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + 1 \\
&\overset{3.2}{=} \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}\left(\mathbf{R}_i \middle| \begin{matrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} + \mathsf{H}\left(\mathbf{R}_i \middle| \begin{matrix} \mathbf{E}' = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E}' = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} \right) + 1 \\
&\overset{3.2}{\leq} \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}\left(\mathbf{R}_i \middle| \begin{matrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} + \mathsf{H}_0\left(\mathbf{R}_i \middle| \begin{matrix} \mathbf{E}' = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E}' = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} \right) + 1 \\
&\overset{3.5}{\leq} \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}\left(\mathbf{R}_i \middle| \begin{matrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} + \lambda^k \cdot \Pr \begin{bmatrix} \mathbf{E}' = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} \right) + 1 \\
&= \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E}' = 1 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} \right) + \lambda^k \cdot \Pr[\mathbf{E}' = 0] + 1 \\
&\overset{3.2}{=} \mathsf{H}(\mathbf{R}_\iota | \mathbf{E}', \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) - \sum_{\widehat{\sigma}, \widehat{U}_H, \widehat{r}_H} \left(\mathsf{H}\left(\mathbf{R}_\iota \middle| \begin{matrix} \mathbf{E}' = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{matrix} \right) \cdot \Pr \begin{bmatrix} \mathbf{E}' = 0 \\ \mathbf{U}_H = \widehat{U}_H \\ \mathbf{r}_H = \widehat{r}_H \\ \boldsymbol{\sigma} = \widehat{\sigma} \end{bmatrix} \right) + \mathsf{negl}(\lambda) + 1 \\
&\leq \mathsf{H}(\mathbf{R}_\iota | \mathbf{E}', \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + \mathsf{negl}(\lambda) + 1 \\
&\overset{3.1.3}{\leq} \mathsf{H}(\mathbf{R}_\iota | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) + \mathsf{negl}(\lambda) + 1.
\end{aligned}$$

Notice that H is a legitimate set of honest parties and $\iota \notin H$, so by Claim 4.1.3,

$$\mathsf{H}(\mathbf{R}_\iota | \mathbf{U}_H, \mathbf{r}_H, \boldsymbol{\sigma}) = O(\log \lambda).$$

We conclude that $\mathsf{H}(\mathbf{R}_i | \boldsymbol{\sigma}) \leq \frac{n}{t} \cdot O(\log \lambda)$.

■

□

4.1 Distributed Sampler CRSs Cannot be Used Twice

We now discuss the first consequence of Theorem 4.1. Suppose that our distribution $\mathcal{D}(\mathbb{1}^\lambda)$ has high min-entropy, i.e. $\mathsf{H}_\infty(\mathcal{D}) = \omega(\log\lambda)$. We observe that the probability that two independent samples from $\mathcal{D}(\mathbb{1}^\lambda)$ collide is negligible. Indeed, denoting the two independent outputs by $\mathbf{R}$ and $\mathbf{R}'$, we have

$$\Pr[\mathbf{R} = \mathbf{R}'] = 2^{-\mathsf{H}_2(\mathcal{D})} \le 2^{-\mathsf{H}_\infty(\mathcal{D})} = 2^{-\omega(\log\lambda)}.$$

We show, however, that if we run a strongly semi-maliciously secure distributed sampler for $\mathcal{D}(\mathbb{1}^\lambda)$ twice using the same CRS, the outputs collide with non-negligible probability. As a consequence, we cannot hope to reuse the same CRS to generate independent looking samples.

Corollary 4.1. *Let $\mathcal{D}(\mathbb{1}^\lambda)$ be an efficiently samplable distribution. Consider a strongly semi-maliciously secure, t-out-of-n distributed sampler protocol for $\mathcal{D}(\mathbb{1}^\lambda)$ with black-box simulation. Suppose that $t/n = \Theta(1)$ and let $\mathbf{R}_i$ and $\mathbf{R}'_i$ denote the outputs of any party P_i in two protocol executions having the same CRS. Then, $\Pr[\mathbf{R}_i = \mathbf{R}'_i] \ge 1/\mathsf{poly}(\lambda)$.*

Proof. By Theorem 4.1, we know that $\mathsf{H}(\mathbf{R}_i|\boldsymbol{\sigma}) = O(\log\lambda)$. Now, we have that

$$\begin{aligned}\Pr[\mathbf{R}_i = \mathbf{R}'_i] &= \sum_{\widehat{\sigma}} \Pr[\boldsymbol{\sigma} = \widehat{\sigma}] \cdot \Pr[\mathbf{R}_i = \mathbf{R}'_i|\boldsymbol{\sigma} = \widehat{\sigma}] \overset{3.7}{=} \sum_{\widehat{\sigma}} \Pr[\boldsymbol{\sigma} = \widehat{\sigma}] \cdot 2^{-\mathsf{H}_2(\mathbf{R}_i|\boldsymbol{\sigma}=\widehat{\sigma})} \\ &\overset{3.2}{\ge} \sum_{\widehat{\sigma}} \Pr[\boldsymbol{\sigma} = \widehat{\sigma}] \cdot 2^{-\mathsf{H}(\mathbf{R}_i|\boldsymbol{\sigma}=\widehat{\sigma})}\end{aligned}$$

We observe that $f(x) := 2^{-x}$ is a convex function, so, by Jensen's inequality

$$\Pr[\mathbf{R}_i = \mathbf{R}'_i] \ge \sum_{\widehat{\sigma}} \Pr[\boldsymbol{\sigma} = \widehat{\sigma}] \cdot 2^{-\mathsf{H}(\mathbf{R}_i|\boldsymbol{\sigma}=\widehat{\sigma})} \ge 2^{-\sum_{\widehat{\sigma}} \Pr[\boldsymbol{\sigma}=\widehat{\sigma}]\cdot\mathsf{H}(\mathbf{R}_i|\boldsymbol{\sigma}=\widehat{\sigma})} \overset{3.2}{=} 2^{-\mathsf{H}(\mathbf{R}_i|\boldsymbol{\sigma})}.$$

Observe that $2^{-\mathsf{H}(\mathbf{R}_i|\boldsymbol{\sigma})} = 1/\mathsf{poly}(\lambda)$. □

4.2 Distributed Sampler CRSs Cannot be Short

A second consequence of Theorem 4.1 is that the CRSs of strongly semi-maliciously secure distributed samplers cannot be short. Specifically, the bit-length of the CRS $|\sigma|$ must be at most $O(\log\lambda)$ bits shorter than $\mathsf{H}_{\mathsf{Yao}}(\mathcal{D})$.

The Yao entropy $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}|\boldsymbol{\sigma})$ must be small. Although Yao's entropy and Shannon's entropy can assume very different values, we prove that if $\mathsf{H}(\mathbf{R}|\boldsymbol{\sigma}) = O(\log\lambda)$, also $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}|\boldsymbol{\sigma}) = O(\log\lambda)$. Indeed, by Corollary 4.1, we know that two distributed sampler executions using the same CRS have colliding outputs with non-negligible probability. We can therefore consider the compressor that on input (R, σ) outputs the empty string and the decompressor that on input σ, runs the distributed sampler using σ as CRS, outputting the result R'. Since there is a $1/\mathsf{poly}(\lambda)$ probability that $R = R'$, we conclude that $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}|\boldsymbol{\sigma}) = O(\log\lambda)$. Observe that we can make the decompressor deterministic using a PRF.

A chain rule for Yao's entropy. To conclude our argument, we show that

$$\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}|\boldsymbol{\sigma}) \ge \mathsf{H}_{\mathsf{Yao}}(\mathbf{R}) - |\boldsymbol{\sigma}|. \tag{2}$$

By the security of distributed samplers in the fully honest case, $\mathbf{R}$ and $\mathcal{D}(\mathbb{1}^\lambda)$ are computationally indistinguishable, so, $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}) = \mathsf{H}_{\mathsf{Yao}}(\mathcal{D})$. From this, we easily deduce that $\mathsf{H}_{\mathsf{Yao}}(\mathcal{D}) - |\boldsymbol{\sigma}| \le O(\log\lambda)$.

We highlight that in [KPW13, Appendix B], Krenn *et al.* proved the chain rule for Yao's entropy, which seems to immediately imply (2). Unfortunately, this is not the case. The idea at the base of their proof is that given a compressor-decompressor pair (c, d) for $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}|\boldsymbol{\sigma})$, we can build a new compressor-decompressor pair for $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R})$ with the same success probability as (c, d). On input R, the new compressor performs a brute-force search for a σ' such that $d(c(R, \sigma'), \sigma') = R$, then it outputs $c(R, \sigma'), \sigma'$. The decompressor instead is identical to d. Since the output size of the new compressor is $|\boldsymbol{\sigma}|$ bits larger than c's, we obtain that $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}|\boldsymbol{\sigma}) \geq \mathsf{H}_{\mathsf{Yao}}(\mathbf{R}) - |\boldsymbol{\sigma}|$. Observe however that if $|\boldsymbol{\sigma}|$ is more than $O(\log \lambda)$, the new compressor does not run in polynomial time. That prevents us from using their result.

We notice that in our setting, the new compressor does not need to perform a brute-force search. Indeed, in order to obtain a σ', it can just feed R to the distributed sampler simulator for the fully honest case and pick the CRS contained in the simulated view. The latter is indistinguishable from the the real CRS used for the generation of R. This allows us to prove the chain rule even if $|\boldsymbol{\sigma}|$ is more than $O(\log \lambda)$.

Corollary 4.2. *Suppose that $t/n = \Theta(1)$. If OWFs exist, the CRS $\boldsymbol{\sigma}$ of a strongly semi-maliciously secure, t-out-of-n distributed sampler for $\mathcal{D}(\mathbb{1}^\lambda)$ with black-box simulation must satisfy $\mathsf{H}_{\mathsf{Yao}}(\mathcal{D}) - |\boldsymbol{\sigma}| \leq O(\log \lambda)$.*

Proof. We start by proving the following claim.

Claim 4.1.5. *In the distributed sampler protocol, for any $i \in [n]$, we have $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i|\boldsymbol{\sigma}) = O(\log \lambda)$.*

Proof of the claim. Consider the following randomised pair of compressor and decompressor:

- On input a pair (R_i, σ), the compressor c outputs the empty string.
- On input σ, the decompressor d runs the distributed sampler protocol in-its-head using σ as CRS. Then, it outputs the result R_i'.

Observe that $\Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] = \Pr[\mathbf{R}_i = \mathbf{R}_i']$. Since $\mathbf{R}_i$ and $\mathbf{R}_i'$ are the outputs of two distributed sampler executions using the same CRS, by Corollary 4.1, we know that

$$\Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] = \frac{1}{\mathsf{poly}(\lambda)}$$

Now, consider the deterministic decompressor d' that performs exactly the same operations as d, but uses a PRF F to generate its randomness, i.e. d' has a random PRF key K hard-coded in its circuit and it generates its randomness by computing $F(K, \sigma)$. By the security of the PRF

$$\Pr[d'(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] \geq \Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] - \mathsf{negl}(\lambda) = \frac{1}{\mathsf{poly}(\lambda)}.$$

Notice that the probability in the first term is also over $\mathbf{K}$. So,

$$\begin{aligned}\frac{1}{\mathsf{poly}(\lambda)} &\leq \Pr[d'(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] = \sum_{\hat{K}} \frac{1}{|\mathbf{K}|} \cdot \Pr[d'(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i | \mathbf{K} = \hat{K}] \\ &\leq \max_{\hat{K}} \Pr[d'(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i | \mathbf{K} = \hat{K}].\end{aligned}$$

Let $\widetilde{K}$ be the value of $\hat{K}$ associated with the maximum, let $d'_{\widetilde{K}}$ be the decompressor having $K = \widetilde{K}$. We have proven that the pair $(c, d'_{\widetilde{K}})$ is successful in compressing and decompressing with probability greater than $1/\mathsf{poly}(\lambda)$. We conclude that $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i|\boldsymbol{\sigma}) = O(\log \lambda)$. ∎

Claim 4.1.6. *In the distributed sampler protocol, for any $i \in [n]$, we have $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i|\boldsymbol{\sigma}) \geq \mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i) - |\boldsymbol{\sigma}|$.*

Proof of the claim. Suppose that $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i|\boldsymbol{\sigma}) \leq k(\lambda)$ for some function $k(\lambda)$. Then, there exists a compressor-decompressor pair (c, d) such that

$$\Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] \geq \frac{2^\ell}{2^k} - \mathsf{negl}(\lambda).$$

We now design a new compressor $\hat{c}$ for $\mathbf{R}_i$: $\hat{c}$ feeds its input R_i to the distributed sampler simulator Sim for the fully-honest case. The latter provides a CRS σ' and the messages of all the parties. The compressor outputs $c(R_i, \sigma')$ as well as σ'.

Let $\boldsymbol{\sigma}$ be the CRS used to generate $\mathbf{R}_i$. By the security of the distributed sampler in the fully honest case, we know that the triple $(\boldsymbol{\sigma}, (\mathbf{U}_j)_{j\in[n]}(\mathbf{R}_j)_{j\in[n]})$ in the protocol is computationally indistinguishable from $(\mathsf{Sim}(1\!\!1^\lambda, \mathbf{R}'), (\mathbf{R}')_{j\in[n]})$ where $\mathbf{R}' \overset{\$}{\leftarrow} \mathcal{D}(1\!\!1^\lambda)$. We conclude that the pairs $(\mathbf{R}_i, \boldsymbol{\sigma})$ and $(\mathbf{R}_i, \boldsymbol{\sigma}')$ are also computationally indistinguishable. As a consequence,

$$\Big|\Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] - \Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}'), \boldsymbol{\sigma}') = \mathbf{R}_i]\Big| = \mathsf{negl}(\lambda).$$

We conclude that

$$\begin{aligned}\Pr[d(\hat{c}(\mathbf{R}_i)) = \mathbf{R}_i] &= \Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}'), \boldsymbol{\sigma}') = \mathbf{R}_i] \geq \Pr[d(c(\mathbf{R}_i, \boldsymbol{\sigma}), \boldsymbol{\sigma}) = \mathbf{R}_i] - \mathsf{negl}(\lambda)\\ &\geq \frac{2^\ell}{2^k} - \mathsf{negl}(\lambda) = \frac{2^{\ell+|\boldsymbol{\sigma}|}}{2^{k+|\boldsymbol{\sigma}|}} - \mathsf{negl}(\lambda).\end{aligned}$$

Observe that $\ell + |\boldsymbol{\sigma}|$ is the output size of $\hat{c}$. Notice also that $\hat{c}$ is not deterministic, however, we can make it so adopting the same technique used in Claim 4.1.5, i.e. by generating its randomness using a PRF. Specifically, consider the deterministic compressor $\hat{c}'$ which has a PRF key K hard-coded in its circuit and generates its randomness by computing $F(K, R_i)$, performing then the same operations as $\hat{c}$. By the security of the PRF, we have that

$$\Big|\Pr[d(\hat{c}'(\mathbf{R}_i)) = \mathbf{R}_i] - \Pr[d(\hat{c}(\mathbf{R}_i)) = \mathbf{R}_i]\Big| = \mathsf{negl}(\lambda).$$

So,

$$\Pr[d(\hat{c}'(\mathbf{R}_i)) = \mathbf{R}_i] \geq \frac{2^{\ell+|\boldsymbol{\sigma}|}}{2^{k+|\boldsymbol{\sigma}|}} - \mathsf{negl}(\lambda).$$

As in the proof of the previous claim, the probability in the first term is also over $\mathbf{K}$. So,

$$\begin{aligned}\frac{2^{\ell+|\boldsymbol{\sigma}|}}{2^{k+|\boldsymbol{\sigma}|}} - \mathsf{negl}(\lambda) \leq &\Pr[d(\hat{c}'(\mathbf{R}_i)) = \mathbf{R}_i] = \sum_{\hat{K}} \frac{1}{|\mathbf{K}|} \cdot \Pr[d(\hat{c}'(\mathbf{R}_i)) = \mathbf{R}_i | \mathbf{K} = \hat{K}]\\ \leq &\max_{\hat{K}} \Pr[d(\hat{c}'(\mathbf{R}_i)) = \mathbf{R}_i | \mathbf{K} = \hat{K}].\end{aligned}$$

Let $\widetilde{K}$ be the value of $\hat{K}$ associated with the maximum, let $\hat{c}'_{\widetilde{K}}$ be the decompressor having $\mathbf{K} = \widetilde{K}$. We have proven that the pair $(\hat{c}'_{\widetilde{K}}, d)$ succeeds in compressing and decompressing $\mathbf{R}_i$ with probability greater than $2^{\ell+|\boldsymbol{\sigma}|}/2^{k+|\boldsymbol{\sigma}|} - \mathsf{negl}(\lambda)$, so $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i) \leq k(\lambda) + |\boldsymbol{\sigma}|$. We conclude that $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i|\boldsymbol{\sigma}) \geq \mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i) - |\boldsymbol{\sigma}|$. ■

By Claims 4.1.5 and 4.1.6, we have $O(\log\lambda) = \mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i|\boldsymbol{\sigma}) \geq \mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i) - |\boldsymbol{\sigma}|$. We notice that, by the security of the distributed sampler in the fully honest case, $\mathbf{R}_i$ is computationally indistinguishable from $\mathcal{D}(1\!\!1^\lambda)$, so, $\mathsf{H}_{\mathsf{Yao}}(\mathbf{R}_i) = \mathsf{H}_{\mathsf{Yao}}(\mathcal{D})$. We conclude that $\mathsf{H}_{\mathsf{Yao}}(\mathcal{D}) - |\boldsymbol{\sigma}| \leq O(\log\lambda)$. □

4.3 Distributed Sampler CRSs Cannot be (too) Nice

The *niceness* of a CRS cannot be defined in a mathematical way. However, informally speaking, we can say that a CRS is nicer than another if it is easier to produce in an MPC setting, e.g. a uniformly random string of bits (i.e. a URS) is simpler to generate than a random RSA modulus of unknown factorisation. Indeed, if we aim for security with abort, we can generate a URS using a simple commit-then-reveal approach. On the other hand, all the state-of-the-art constructions for the generation of RSA moduli rely on rejection sampling: first the parties generate secret-shared (or encrypted) candidate primes p and q, they multiply them and apply expensive (bi)primality tests on the secret-shared data [FLOP18, HMR+19, CCD+20]. After sufficiently many trials (the number depends on the size of the modulus), the players obtain a valid RSA modulus with high probability. This results in rather complex protocols.

In the previous sections, we have seen that the CRS of distributed samplers cannot be used more then once and cannot be short. These facts suggest that directly encoding a sample from $\mathcal{D}(1\!\!1^\lambda)$ in a CRS is probably better than relying on a distributed sampler protocol for $\mathcal{D}(1\!\!1^\lambda)$. At least in the first case, the parties spare one round of interaction. But what if the CRS used by the distributed sampler is nicer than any encoding of $\mathcal{D}(1\!\!1^\lambda)$? We prove that this cannot happen as it is always possible to non-interactively generate samples from $\mathcal{D}(1\!\!1^\lambda)$ using only distributed sampler CRSs and public random coins.

Non-interactive generation of samples from $\mathcal{D}(1\!\!1^\lambda)$ using a distributed sampler CRS and random bits. In this section, we observe that a distributed sampler for $\mathcal{D}(1\!\!1^\lambda)$ allows us to construct an efficient deterministic function De that maps pairs consisting of a distributed sampler CRS σ and uniformly random bits r into values R that are computationally indistinguishable from $\mathcal{D}(1\!\!1^\lambda)$. This algorithm trivially outputs the result of the distributed sampler using σ as CRS and r as randomness for the parties. The interesting fact is that if the distributed sampler is strongly semi-maliciously secure with black-box simulation, the algorithm is efficiently invertible with non-negligible probability. Specifically, we prove that there exists an efficient PPT algorithm test, that outputs 1 with $1/\mathsf{poly}(\lambda)$ probability when run over a sample $R \xleftarrow{\$} \mathcal{D}(1\!\!1^\lambda)$. Moreover, if this event occurs, it is possible to efficiently find (using an algorithm En) a pair (σ, r) that looks random over $\mathsf{De}^{-1}(R)$.

In other words, if we provide the parties with a distributed sampler CRS σ and public uniformly random bits r, the parties can obtain a sample R from $\mathcal{D}(1\!\!1^\lambda)$ without any interaction. Furthermore, with $1/\mathsf{poly}(\lambda)$ probability, (σ, r) reveals no information in addition to what can be already inferred from R. Finally, the honest parties can tell when (σ, r) reveals too much information, having therefore the opportunity to reject R and restart.

Biases in the distribution. Notice that the selective rejection biases the distribution of the output. However, any cryptographic protocol basing its security on R will remain secure even if R is sampled according to the biased distribution. Indeed, at least a polynomial fraction of the samples is not rejected. If the protocol was insecure in the new setting, there would be a non-negligible probability of sampling a bad R even in the original protocol, which would therefore be insecure.

Why is De *invertible with non-negligible probability?* Our idea is based on the fact that in a strongly semi-malicious distributed sampler, $\mathsf{H}(\mathbf{R}|\boldsymbol{\sigma})$ is small. In other words, the CRS of the protocol describes the output with high precision. Obtaining the exact CRS used for the generation of $\mathbf{R}$ is usually hard, however, the simulator for the fully-honest case can provide us with a functionally equivalent object.

In order to completely describe $\mathbf{R}$, we need to extract also randomness $\mathbf{r}$ for the parties, so that, using $\mathbf{r}$ in conjunction with the CRS, we obtain $\mathbf{R}$. Unfortunately, the simulator cannot provide much help here. Indeed, given $(\boldsymbol{\sigma}', (\mathbf{U}_i)_{i\in[n]}) \xleftarrow{\$} \mathsf{Sim}(1\!\!1^\lambda, \mathbf{R})$, it is usually hard to extract the randomness $\mathbf{r}$ used to generate $(\mathbf{U}_i)_{i\in[n]}$. Actually, such $\mathbf{r}$ might not even exist. Luckily, in Corollary 4.1, we have proven that two executions of a strongly semi-maliciously secure distributed sampler using the same CRS have colliding outputs with $1/\mathsf{poly}(\lambda)$ probability. So, if we run the distributed sampler protocol again using $\boldsymbol{\sigma}'$ as CRS, we obtain $\mathbf{R}$ again with non-negligible probability. Clearly, in the new execution, the value of $(\mathbf{U}_i)_{i\in[n]}$ has probably changed, however, this time we know the randomness $\mathbf{r}'$ used to generate the messages.

On average invertibility. We observe that the probability of succeeding in inverting De is also over the outcome of $\mathbf{R}$. In other words, we just know that the *average* probability of inverting $\mathbf{R}$ is $1/\mathsf{poly}(\lambda)$. That does not mean that the overwhelming majority of values of $\mathbf{R}$ is efficiently invertible: if $\mathbf{R}$ assumes an unlucky value, we can try to invert as many times as we want without any hope of succeeding. We could prove, however, that there always exists a polynomial fraction of the space of events for which $\mathbf{R}$ is easy to invert.

We also noticed that it is possible to efficiently test if a value R is easy to invert or not. Indeed, we can just try to invert it many times, if the success frequency is lower than a certain threshold, we can reject R, otherwise, we accept it. We used the Chernoff bound to find the threshold and the maximum number of inversion attempts. In particular, we needed test to succeed with at least $1/\mathsf{poly}(\lambda)$ probability.

The special case of URSs. As a corollary of the result described in this section, if the distributed sampler uses a URS, it is possible to securely generate samples from $\mathcal{D}(1\!\!1^\lambda)$ using public random coins only. In particular, in the random oracle model, the parties can securely sample from $\mathcal{D}(1\!\!1^\lambda)$ without interacting and without needing any CRS.

Corollary 4.3. *Suppose that there exists a strongly semi-maliciously secure, t-out-of-n distributed sampler for $\mathcal{D}(1\!\!1^\lambda)$ with black-box simulation. Suppose that $t/n = \Theta(1)$ and let $\mathsf{CRS}(1\!\!1^\lambda)$ be the algorithm used to generate the CRS of the protocol. Then, there exist a deterministic polynomial algorithm* De *and PPT algorithms* En *and* test *such that*

– $\mathsf{De}\big(\mathsf{CRS}(1\!\!1^\lambda), \mathcal{U}(1\!\!1^\lambda)\big) \sim_c \mathcal{D}(1\!\!1^\lambda)$

THE GAME $\mathcal{G}_{\mathcal{A}}^{\mathsf{Inv}}(\mathbb{1}^\lambda)$

The challenger performs the following operations

1. $b \overset{\$}{\leftarrow} \{0,1\}$
2. $\sigma \overset{\$}{\leftarrow} \mathsf{CRS}(\mathbb{1}^\lambda)$, $r \overset{\$}{\leftarrow} \mathcal{U}(\mathbb{1}^\lambda)$
3. $R \leftarrow \mathsf{De}(\sigma, r)$
4. If $\mathsf{test}(R) = 0$, go back to step 2.
5. If $b = 0$, provide the adversary with (σ, r), otherwise provide it with $\mathsf{En}(R)$.

The adversary wins if it terminates its execution outputting b.

Fig. 2: Invertibility game

– $\Pr[\mathsf{test}(\mathcal{D}(\mathbb{1}^\lambda)) = 1] \geq 1/\mathsf{poly}(\lambda)$
– *No PPT adversary can win the game* $\mathcal{G}_{\mathcal{A}}^{\mathsf{Inv}}(\mathbb{1}^\lambda)$ *(see Fig. 2) with non-negligible advantage.*

Proof. We start by defining the algorithm De, which on input σ and random string r, runs the distributed sampler protocol using σ as CRS and r as randomness for the parties, outputting the result of party P_1.

Claim 4.1.7. $\mathsf{De}\big(\mathsf{CRS}(\mathbb{1}^\lambda), \mathcal{U}(\mathbb{1}^\lambda)\big) \sim_c \mathcal{D}(\mathbb{1}^\lambda)$

Proof of the claim. We observe that $\mathbf{R} := \mathsf{De}\big(\mathsf{CRS}(\mathbb{1}^\lambda), \mathcal{U}(\mathbb{1}^\lambda)\big)$ is distributed exactly as the distributed sampler output. By the security of the distributed sampler, we conclude that $\mathbf{R} \sim_c \mathcal{D}(\mathbb{1}^\lambda)$. ■

We now define the PPT algorithm Inv as follows: Inv feeds the input R to the distributed sampler simulator for the fully honest case. In this way, it obtains a fake CRS σ' and messages $(U_i)_{i\in[n]}$. Finally, Inv picks a uniformly random string r and outputs (σ', r).

Claim 4.1.8. *Let* $\mathbf{R}$ *denote a sample from* $\mathcal{D}(\mathbb{1}^\lambda)$*. Then, there exists a polynomial* $q(\lambda)$ *such that*

$$\Pr[\mathsf{De}(\mathsf{Inv}(\mathbf{R})) = \mathbf{R}] \geq \frac{1}{q(\lambda)}$$

Proof of the claim. Let $(\boldsymbol{\sigma}', \mathbf{r}) := \mathsf{Inv}(\mathbf{R})$. By the security of distributed samplers, we know that $(\boldsymbol{\sigma}', \mathbf{R})$ is computationally indistinguishable from $(\boldsymbol{\sigma}, \mathsf{De}(\boldsymbol{\sigma}, \mathbf{r}))$ where $\boldsymbol{\sigma} \overset{\$}{\leftarrow} \mathsf{CRS}(\mathbb{1}^\lambda)$ and $\mathbf{r}$ is uniformly random. We conclude that for $\mathbf{r}$ and $\mathbf{r}'$ independent and uniformly random

$$\big(\boldsymbol{\sigma}, \mathsf{De}(\boldsymbol{\sigma}, \mathbf{r}'), \mathsf{De}(\boldsymbol{\sigma}, \mathbf{r})\big) \sim_c \big(\boldsymbol{\sigma}', \mathbf{R}, \mathsf{De}(\boldsymbol{\sigma}', \mathbf{r})\big) \tag{3}$$

By Corollary 4.1, we know that

$$\Pr[\mathsf{De}(\boldsymbol{\sigma}, \mathbf{r}') = \mathsf{De}(\boldsymbol{\sigma}, \mathbf{r})] \geq \frac{1}{\mathsf{poly}(\lambda)}$$

We conclude that, by (3),

$$\Pr[\mathsf{De}(\mathsf{Inv}(\mathbf{R})) = \mathbf{R}] = \Pr[\mathsf{De}(\boldsymbol{\sigma}', \mathbf{r}') = \mathbf{R}] \geq \frac{1}{\mathsf{poly}(\lambda)} - \mathsf{negl}(\lambda) \geq \frac{1}{\mathsf{poly}(\lambda)}$$

■

We now define the algorithm test as follows: on input R, test checks if $\mathsf{De}(\mathsf{Inv}(R)) = R$ for $4\lambda \cdot q(\lambda)$ times. If the equation is satisfied less than λ times, test outputs 0, otherwise it output 1.

In a similar way, we define En: on input R, En computes $(\sigma', r) \overset{\$}{\leftarrow} \mathsf{Inv}(R)$ and checks if $\mathsf{De}(\sigma', r) = R$. If that is the case, it outputs (σ', r). Otherwise, it repeats the operation. If the procedure fails for more than $8\lambda \cdot q(\lambda)$, En outputs $\perp$.

Claim 4.1.9.
$$\Pr[\mathsf{test}(\mathcal{D}(\mathbb{1}^\lambda)) = 1] \geq \frac{1}{8q(\lambda)}$$

Proof of the claim. For every value x, we define $p_x := \Pr[\mathsf{De}(\mathsf{Inv}(x)) = x]$. Let $\mathbf{R}$ be a random variable denoting a sample from $\mathcal{D}(\mathbb{1}^\lambda)$ and notice that $p_{\mathbf{R}}$ is a random variable. By Claim 4.1.8, we know that

$$\mathbb{E}[p_{\mathbf{R}}] = \sum_x \Pr[\mathbf{R} = x] \cdot p_x = \Pr[\mathsf{De}(\mathsf{Inv}(\mathbf{R})) = \mathbf{R}] \geq \frac{1}{q(\lambda)}$$

Since $0 \leq p_x \leq 1$, we have that $\mathbb{E}[p_{\mathbf{R}}^2] \leq \mathbb{E}[p_{\mathbf{R}}]$, so, by the Paley-Zygmund inequality,

$$\Pr\Big[p_{\mathbf{R}} \geq \frac{1}{2q(\lambda)}\Big] \geq \Pr\Big[p_{\mathbf{R}} > \frac{1}{2}\mathbb{E}[p_{\mathbf{R}}]\Big] \geq \frac{1}{4} \cdot \frac{\mathbb{E}[p_{\mathbf{R}}]^2}{\mathbb{E}[p_{\mathbf{R}}^2]} \geq \frac{1}{4} \cdot \mathbb{E}[p_{\mathbf{R}}] \geq \frac{1}{4q(\lambda)}$$

Define now $\Omega_{\mathsf{good}} = \{x | p_x \geq \frac{1}{2q(\lambda)}\}$. Suppose now that $x \in \Omega_{\mathsf{good}}$. If we run the check $\mathsf{De}(\mathsf{Inv}(x)) = x$ for $8\lambda \cdot q(\lambda)$ times, by the Chernoff bound, we know that it succeeds more than

$$\frac{1}{2} \cdot 4\lambda q(\lambda) \cdot p_x \geq \lambda$$

times with overwhelming probability. So if $x \in \Omega_{\mathsf{good}}$, $\mathsf{test}(x) = 1$ with overwhelming probability. We conclude that

$$\begin{aligned}\Pr[\mathsf{test}(\mathbf{R}) = 1] &\geq \Pr[\mathsf{test}(\mathbf{R}) = 1 | \mathbf{R} \in \Omega_{\mathsf{good}}] \cdot \Pr[\mathbf{R} \in \Omega_{\mathsf{good}}] \\ &\geq \min_{x \in \Omega_{\mathsf{good}}} \Pr[\mathsf{test}(x) = 1] \cdot \Pr[\mathbf{R} \in \Omega_{\mathsf{good}}] \geq \frac{1}{8q(\lambda)}\end{aligned}$$

■

Claim 4.1.10. *No PPT adversary can win the game $\mathcal{G}_{\mathcal{A}}^{\mathsf{Inv}}(\mathbb{1}^\lambda)$ (see Fig. 2) with non-negligible advantage.*

Proof of the claim. As in the previous claim, let $p_x := \Pr[\mathsf{De}(\mathsf{Inv}(x)) = x]$. Define now $\Omega_{\mathsf{bad}} = \{x | p_x \leq \frac{1}{8q(\lambda)}\}$. Suppose that $x \in \Omega_{\mathsf{bad}}$. If we run the check $\mathsf{De}(\mathsf{Inv}(x)) = x$ for $4\lambda \cdot q(\lambda)$ times, by the Chernoff bound, we know that it succeeds less than

$$2 \cdot 4\lambda q(\lambda) \cdot p_x \leq \lambda$$

times with overwhelming probability. So, if $x \in \Omega_{\mathsf{bad}}$, $\mathsf{test}(x) = 0$ with overwhelming probability.

Let $\mathbf{R} \xleftarrow{\$} \mathcal{D}(\mathbb{1}^\lambda)$. We observe that

$$\Pr[\mathsf{En}(\mathbf{R}) = \bot | \mathsf{test}(\mathbf{R}) = 1] \leq \Pr[\mathbf{R} \in \Omega_{\mathsf{bad}} | \mathsf{test}(\mathbf{R}) = 1] + \Pr[\mathsf{En}(\mathbf{R}) = \bot | \mathbf{R} \notin \Omega_{\mathsf{bad}}].$$

We know that

$$\begin{aligned}\Pr[\mathbf{R} \in \Omega_{\mathsf{bad}} | \mathsf{test}(\mathbf{R}) = 1] &\leq \frac{\Pr[\mathsf{test}(\mathbf{R}) = 1 | \mathbf{R} \in \Omega_{\mathsf{bad}}]}{\Pr[\mathsf{test}(\mathbf{R}) = 1]} \\ &\leq 8q(\lambda) \cdot \max_{x \in \Omega_{\mathsf{bad}}} \Pr[\mathsf{test}(x) = 1] = \mathsf{negl}(\lambda).\end{aligned}$$

Furthermore,
$$\Pr[\mathsf{En}(\mathbf{R}) = \bot | \mathbf{R} \notin \Omega_{\mathsf{bad}}] \geq \min_{x \notin \Omega_{\mathsf{bad}}} \Pr[\mathsf{En}(x) = \bot].$$

Notice that for every $x \notin \Omega_{\mathsf{bad}}$, $p_x > \frac{1}{8q(\lambda)}$. Observe also that En tries to invert the input up to $8\lambda \cdot q(\lambda) > \lambda / p_x$ times, so, with overwhelming probability $\mathsf{En}(x) \neq \bot$. We conclude that $\Pr[\mathsf{En}(\mathbf{R}) = \bot | \mathsf{test}(\mathbf{R}) = 1] = \mathsf{negl}(\lambda)$.

Now, suppose that $\mathsf{En}(\mathbf{R}) = (\boldsymbol{\sigma}', \mathbf{r}') \neq \bot$. We recall that $\boldsymbol{\sigma}'$ is obtained by running $\mathsf{Sim}(\mathbb{1}^\lambda, \mathbf{R})$, so by the security of distributed samplers $(\mathbf{R}, \boldsymbol{\sigma}') \sim_c (\mathsf{De}(\boldsymbol{\sigma}, \mathbf{r}), \boldsymbol{\sigma})$ where $\boldsymbol{\sigma} \xleftarrow{\$} \mathsf{CRS}(\mathbb{1}^\lambda)$ and $\mathbf{r} \xleftarrow{\$} \mathcal{U}(\mathbb{1}^\lambda)$. We also know that $\mathbf{r}'$ is random conditioned on satisfying $\mathsf{De}(\boldsymbol{\sigma}', \mathbf{r}') = \mathbf{R}$. In other words, $(\mathbf{R}, \boldsymbol{\sigma}', \mathbf{r}')$ is computationally indistinguishable from $(\mathsf{De}(\boldsymbol{\sigma}, \mathbf{r}), \boldsymbol{\sigma}, \mathbf{r})$. We conclude our proof observing that $(\mathbf{R}, \mathsf{En}(\mathbf{R})) \sim_c (\mathsf{De}(\boldsymbol{\sigma}, \mathbf{r}), \mathsf{En}(\mathsf{De}(\boldsymbol{\sigma}, \mathbf{r})))$. ■

□

Acknowledgments

The authors would like to thank Ivan Damgård, Jesper Buus Nielsen and Sophia Yakoubov for their feedback and support.

D.A. and P.S. were supported by the Aarhus University Research Foundation (AUFF) and the Independent Research Fund Denmark (DFF) (grant DFF-0165-00107B "C3PO"). D.A was supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation). M.O. was supported by MOE tier 2 grant - T2EP20223-0024, and NRF award NRF- NRFI09-0005.

References

ADIN24. Damiano Abram, Jack Doerner, Yuval Ishai, and Varun Narayanan. Constant-Round Simulation-Secure Coin Tossing Extension with Guaranteed Output. In *EUROCRYPT 2024*. Springer, 2024.

AOS23. Damiano Abram, Maciej Obremski, and Peter Scholl. On the (im)possibility of distributed samplers: Lower bounds and party-dynamic constructions. Cryptology ePrint Archive, Paper 2023/863, 2023.

ASY22. Damiano Abram, Peter Scholl, and Sophia Yakoubov. Distributed (Correlation) Samplers: How to Remove a Trusted Dealer in One Round. In *EUROCRYPT 2022*. Springer, 2022.

AWZ23. Damiano Abram, Brent Waters, and Mark Zhandry. Security-Preserving Distributed Samplers: How to Generate any CRS in One Round without Random Oracles. In *CRYPTO 2023*. Springer, 2023.

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

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS 2012*. ACM, January 2012.

Bd94. Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *EUROCRYPT'93*, LNCS. Springer, Heidelberg, May 1994.

BMO90. Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. The (true) complexity of statistical zero knowledge. In *22nd ACM STOC*. ACM Press, May 1990.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*. IEEE Computer Society Press, October 2001.

CCD$^+$20. Megan Chen, Ran Cohen, Jack Doerner, Yashvanth Kondi, Eysa Lee, Schuyler Rosefield, and abhi shelat. Multiparty generation of an RSA modulus. In *CRYPTO 2020, Part III*, LNCS. Springer, Heidelberg, August 2020.

CKL03. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT 2003*, LNCS. Springer, Heidelberg, May 2003.

DHRW16. Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO 2016, Part III*, LNCS. Springer, Heidelberg, August 2016.

DLN$^+$04. Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for np and spooky interactions. Unpublished manuscript, 2004.

FLOP18. Tore Kasper Frederiksen, Yehuda Lindell, Valery Osheter, and Benny Pinkas. Fast distributed RSA key generation for semi-honest and malicious adversaries. In *CRYPTO 2018, Part II*, LNCS. Springer, Heidelberg, August 2018.

GK90. Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. In *Automata, Languages and Programming*, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

HIJ$^+$17. Shai Halevi, Yuval Ishai, Abhishek Jain, Ilan Komargodski, Amit Sahai, and Eylon Yogev. Non-interactive multiparty computation without correlated randomness. In *ASIACRYPT 2017, Part III*, LNCS. Springer, Heidelberg, December 2017.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, (4), 1999.

HLR07. Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *EUROCRYPT 2007*, LNCS. Springer, Heidelberg, May 2007.

HMR$^+$19. Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, Tomas Toft, and Angelo Agatino Nicolosi. Efficient RSA key generation and threshold paillier in the two-party setting. *Journal of Cryptology*, (2), April 2019.

HV16. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. What security can we achieve within 4 rounds? In *SCN 16*, LNCS. Springer, Heidelberg, August / September 2016.

KPW13. Stephan Krenn, Krzysztof Pietrzak, and Akshay Wadia. A counterexample to the chain rule for conditional HILL entropy - and what deniable encryption has to do with it. In *TCC 2013*, LNCS. Springer, Heidelberg, March 2013.

Ore87. Yair Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs (extended abstract). In *28th FOCS*. IEEE Computer Society Press, October 1987.

OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT 2021, Part I*, LNCS. Springer, Heidelberg, October 2021.

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, LNCS. Springer, Heidelberg, August 2008.

Rey11. Leonid Reyzin. Some notions of entropy for cryptography - (invited talk). In *ICITS 11*, LNCS. Springer, Heidelberg, May 2011.

Sha48. C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. J.*, 27:623–656, 1948.

Yao82. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*. IEEE Computer Society Press, November 1982.