Semi-Honest Secure Multiparty Computation Can Be Insecure by Using Secure Pseudorandom Generators

Koji Nuida[1,2]

[1] The University of Tokyo, Japan
(nuida@mist.i.u-tokyo.ac.jp)
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan

July 20, 2018

Abstract

It is widely understood that we are just human beings rather than being almighty; hence using ideally random numbers in practice, as supposed in usual theoretical designs of cryptographic protocols, is beyond our ability or at least too expensive. For this point, a standard solution in implementation is to use secure pseudorandom generators (PRGs); ordinary cryptographers' intuition tells that the security of cryptographic protocols should not be lost when applying a secure PRG *though no general proof for this is known*. In this paper, as opposed to the intuition, we give two examples (under certain, different computational assumptions) of a pair of a secure two-party computation protocol in the semi-honest model (one of which is essentially a practical protocol proposed in ACM CCS 2013, not just an artificially constructed one) and a secure PRG satisfying that *the security is lost when the PRG is applied*. This phenomenon comes mainly from the fact that, in the security model for two-party protocols the seed for a PRG will be visible by a corrupted party him/herself, while the security notion for PRGs assumes that the seed is not visible. On the other hand, as an affirmative result, we give a sufficient condition for a two-party protocol and a PRG to ensure that the security is preserved when the PRG is applied.

1 Introduction

In cryptography, the gap between the ideal randomness assumed in the theoretical models and the non-ideal randomness in the real world is always a major worrying problem. For example, Heninger et al. revealed in 2012 [9] that, a surprisingly large part of TLS and SSH servers in the world at that time had serious vulnerability caused by inappropriate generation of random cryptographic keys. In order to avoid such a crucial vulnerability, an honest and earnest cryptographic engineer would like to implement a protocol by following the description of the protocol in the document or the original academic paper as faithfully as possible. However, there is a problem for such an engineer that, though the theoretical design of a cryptographic protocol usually assumes the use of perfectly random numbers, practically generating such perfectly random numbers is even beyond the art of human being or at least requires much expensive cost.

A standard solution for such a problem of randomness generation (at least for computational security settings rather than information-theoretic security) is to use a cryptographically secure *pseudorandom generator* (*PRG*) that stretches a short random seed to generate a long pseudorandom string. In fact, the standard formalization of security notion for a cryptographic PRG intends to be well-suited to any "natural" security notion for cryptographic protocols. As a result, it is an ordinary cryptographer's daily-life intuition that, if a cryptographic protocol is secure, then it *should* remain secure when a secure PRG is applied to generate the internal randomness for the protocol.

Nevertheless, the aim of the present paper is to revisit the seemingly reasonable intuition that a secure PRG should preserve the security of a protocol. Our starting point is that there have been no formal *proofs*

showing that a secure PRG preserves the security of an arbitrary cryptographic protocol. We emphasize that we do NOT claim here that the use of cryptographic PRGs is unreasonable at all in implementing cryptographic protocol; using secure PRGs would indeed preserve the security in most of the cryptographic situations. However, one of the main results of this paper still reveals that, there is a cryptographic situation where the security of the original protocol is *not* preserved when a secure PRG is used.

1.1 Our Results

In this paper, under a certain computational assumption, we show the following: there exists a pair of a computationally *secure* two-party computation protocol in the semi-honest model and a *secure* PRG satisfying that, when a party for the protocol uses the PRG to generate the party's internal random tape, the resulting protocol falls into an *insecure* protocol. We give two such examples in Sections 4 and 5 under different computational assumptions. While the computational assumption in Section 4 is relatively simple, the protocol in the example of Section 5 is more natural than that of Section 4. In fact, the protocol in Section 5 is essentially an oblivious transfer protocol proposed by Asharov et al. in ACM CCS 2013 [1].

Here we give an intuitive explanation of such a "counter-intuitive" insecurity for a PRG-based protocol. First we recall that, a two-party computation protocol enables two parties endowed with their local inputs to compute a function from the two local inputs, in a way that each party's local input (except information implied by the function value itself) is kept secret against the other party during the interactive computation. The security notion of such a protocol against a party in the semi-honest model is described by using a simulator for the party; the protocol is regarded as secure if everything obtained by the party during a protocol execution can be simulated from the party's local input and output only. The important point here is that, the party's random tape itself is also included in the party's "view" during the protocol execution, which has to be simulated by the simulator in a security proof. This convention for the security definition reflects the fact that the random tape is stored in the party's own device, therefore a corrupted party can also utilize this random tape for extracting some information on the other party's secret. Now by the same observation, when the party uses a PRG, the security proof has to suppose that the seed for the PRG (stored in the party's device as well) used during a protocol execution can also be seen by the corrupted party. However, this visibility of the seed conflicts with the security notion of PRGs, the latter assuming that the seed is not visible by the distinguisher for the PRG. Hence, in general, the security of the PRG is not sufficient for ensuring that the security of a two-party protocol is preserved when the PRG is used.

On the other hand, as an affirmative result, in this paper we also give (in Section 6) a sufficient condition for a two-party protocol and a PRG to ensure that the protocol remains secure when a party uses the PRG to generate the internal random tape. To explain the condition, first we observe that, the current formulation of the security for a two-party protocol (in the semi-honest model) allows a security proof using a simulator that generates the party's view except the party's random tape first and then adjusts the random tape as a function of the other part of the view. This is in fact reverse to the chronological order of a real protocol execution where the party's random tape is sampled first and the behavior of the protocol depends on the content of the random tape. Based on the observation, we introduce the following definition: a simulator in a security proof is said to be *with raw random tape* if the simulator chooses the party's simulated random tape first and then generates the remaining part of the party's simulated view depending on the chosen random tape. Our affirmative result in this paper also requires additional condition on the simulator that the output of the simulator is *statistically* (not just computationally) close to the party's view in a real protocol execution. Assuming the "raw random tape" and the "statistical" conditions for the simulator, our result shows that the use of a PRG preserves the security of the two-party protocol provided the output distribution of the PRG with uniformly random seed has sufficiently large min-entropy (e.g., the PRG has logarithmic stretch in the case where the PRG is injective).

We give remarks on the reasons of the two conditions for the simulator in our affirmative result mentioned above. First we consider the "raw random tape" condition. From a purely technical viewpoint, our result in Section 4 would suggest the necessity of the condition as the counterexample in Section 4 lacks this condition while satisfying the other "statistical" condition. Moreover, we can also give the following intuitive explanation. Suppose that the simulator $\mathcal{S}$ in the security proof of a given protocol is not with raw random

tape and the party's random tape in the output of $\mathcal{S}$ is a function, denoted here by F, of the other part of the party's simulated view. Given a secure PRG $\mathcal{R}$, we try to construct a simulator $\widetilde{\mathcal{S}}$ for the party in the protocol combined with the PRG $\mathcal{R}$. Now $\widetilde{\mathcal{S}}$ has to simulate the seed r of the PRG $\mathcal{R}$. On the other hand, $\widetilde{\mathcal{S}}$ has also to simulate the party's view v except the random tape, and this would have to be done by executing the original simulator $\mathcal{S}$. But in this case, the seed r should be sampled with the constraint $\mathcal{R}(r) = F(v)$, which looks difficult as the secure PRG $\mathcal{R}$ is in general hard to invert. Due to the observation, the "raw random tape" condition seems fairly crucial when developing a similar affirmative result.

Secondly, we consider the "statistical" condition for the simulator. Again, our result in Section 5 would suggest the technical necessity of the condition as the counterexample in Section 5 lacks this condition while satisfying the other "raw random tape" condition. Moreover, the aforementioned visibility of the seed for the PRG would make it difficult to develop such an affirmative result by utilizing the computational security properties of the building blocks. In fact, the proof of our affirmative result is information-theoretic rather than cryptographic, as it is based on information-theoretic properties such as the statistical closeness of the simulator and the min-entropy of the PRG. It will be an interesting challenge to improve our affirmative result by utilizing some cryptographic properties rather than information-theoretic ones.

Finally, we give a remark on the standpoint of the results in this paper. We emphasize that, the negative results in this paper do not claim that a two-party protocol will be concretely broken when a practical PRG is used (in fact, the PRGs for our results in Sections 4 and 5 are very artificial) but just point out the lack of a theoretical security proof under the use of a PRG. This situation would have a flavor somewhat similar to the case of security proofs in the random oracle model. Although the random oracle model is just a theoretical approximation and a real hash function is never a random oracle, the security proofs for practical protocols based on the random oracle model are regarded as meaningful at least to some extent. Similarly, even after the negative results of this paper, security proofs for two-party protocols (in the semi-honest model) using ideal randomness would still have meaning at least to some extent when PRGs are applied to those protocols. But at the same time, we emphasize that recognizing the lack of a fully rigorous security proof in the case of two-party protocols combined with PRGs (revealed in this paper) would be as significant as recognizing that the random oracle assumption is never achieved completely in practical protocols.

1.2 Related Work

One may feel that the topic of the present paper seems to be related to some other topics concerning non-ideal randomness in cryptography, such as cryptography based on so-called "imperfect randomness" (e.g., [4, 5]) and the security issues caused by "backdoored PRGs" (e.g., [2, 3]). But actually, the former topic above mainly deals with randomness that is significantly far from being ideal; in contrast, the present paper focuses on the use of randomness that is significantly close to ideal. On the other hand, the latter topic above studies the problem of the use of maliciously (and secretly) designed PRGs; while the main concern of the present paper originates from the practical impossibility of implementing the ideal randomness even if an engineer is honest and makes a best effort. Hence our problem setting is significantly different.

In the paper [12, 13] of Lindell, Nissim, and Orlandi, they gave feasibility results on some classes of functionality in a certain enhanced ("size-hiding") two-party computation protocol under the semi-honest model. At the same time, they also gave an infeasibility result (Theorem 5.7 in [13]) on such a protocol for another class of functionality, but the security model here is different in a way that an adversarial (semi-honest) party is assumed to be *deterministic*. A similar situation also appeared in a recent paper by Shinagawa et al. [15]. In those papers, the authors seemed to had tried to prove the infeasibility result under the semi-honest model by first establishing a kind of lower bounds for "overall complexity" of such protocols that involves the randomness complexity as well, and then cancelling out the effect of the randomness complexity from the overall complexity by replacing the ideal randomness with an output of a computationally secure PRG of sufficiently large stretch. However, our counterexample in the paper shows that such a strategy must fail; namely, when one assumes for contrary the existence of a secure protocol with overall complexity *minus randomness complexity* being larger than a given lower bound and then tries to deduce a contradiction by cancelling out the randomness complexity by replacing the ideal randomness with a PRG, it is *not* guar-

anteed in general that the resulting protocol with PRG is still secure. We note that Hazai and Zarosim in their recent paper [8] already mentioned this problem in such a general strategy to cancel out the effect of randomness complexity when developing a kind of lower bounds for complexity of two-party protocols. However, they did not give a concrete example (as in the present paper) that the use of even secure PRGs compromises the security of a two-party protocol.

On the other hand, in [10], Hubácek and Wichs proposed a kind of secure two-party computation protocol in the semi-honest model, and also gave a lower bound for communication complexity that seemingly excludes even their own protocol but is actually established only for the case of *deterministic* adversaries. They also clearly mentioned that their protocol is an example of the phenomenon where a party's randomness affects security of the other party's secret input. However, their alternative situation of a deterministic adversarial party is far from the original situation with ideal randomness, in contrast to our counterexample where an adversarial party uses a secure PRG.

2 Preliminaries

2.1 Basic Notations and Settings

In this paper, we write $\{0,1\}^* = \bigcup_{n\geq 0}\{0,1\}^n$ and denote the set of positive integers by $\mathbb{Z}_{>0}$. We say that a function $\varepsilon(\lambda) \in [0,1]$ of an integer $\lambda \geq 1$ is *negligible* in λ, if for any $k \in \mathbb{Z}_{>0}$, there exists a $\lambda_0 \in \mathbb{Z}_{>0}$ satisfying $\varepsilon(\lambda) < \lambda^{-k}$ for any $\lambda > \lambda_0$. An anonymous negligible function is sometimes denoted by $\mathsf{negl}(\lambda)$.

For a probability distribution D, we write $a \leftarrow\!\!\!\prec D$ to indicate that the element a is chosen according to the distribution D. Let $U[X]$ denote the uniform distribution on a set X. We use a notation of a form $[F(r)\colon \mathcal{R}]$ to signify a random variable $F(r)$ where r follows the probability distribution specified by the term $\mathcal{R}$. For example, $[bb\colon b \leftarrow\!\!\!\prec U[\{0,1\}]]$ equals $U[\{00,11\}]$. For two probability distributions X and Y over a (finite) set Z, their *statistical distance* $\Delta(X,Y)$ is defined by

$$\Delta(X,Y) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{z\in Z}|\Pr[z \leftarrow\!\!\!\prec X] - \Pr[z \leftarrow\!\!\!\prec Y]| = \max_{E\subseteq Z}\left(\Pr_{z\leftarrow\!\!\!\prec X}[z\in E] - \Pr_{z\leftarrow\!\!\!\prec Y}[z\in E]\right) \ .$$

It is known that we have $\Delta(f(X,D), f(Y,D)) \leq \Delta(X,Y)$ for any function f and any probability distribution D independent of X and Y. We also present the following properties for statistical distances:

Lemma 1. *Let p,q be two distinct λ-bit primes, i.e., $p,q \in [2^{\lambda-1}, 2^\lambda - 1]$, and let $N = pq$. Moreover, we set $U = U[\mathbb{Z}/N\mathbb{Z}]$ and $U' = U[(\mathbb{Z}/N\mathbb{Z})^\times]$. Then we have $\Delta(U,U') < 2^{-(\lambda-2)}$. Hence we also have $\Delta(f(U),U') < 2^{-(\lambda-2)}$ for any function f with domain $\mathbb{Z}/N\mathbb{Z}$ that is identical on the subset $(\mathbb{Z}/N\mathbb{Z})^\times$.*

Proof. By the property of the statistical distance, we have

$$\begin{aligned}\Delta(U,U') = \frac{|(\mathbb{Z}/N\mathbb{Z}) \setminus (\mathbb{Z}/N\mathbb{Z})^\times|}{N} &= \frac{N-(p-1)(q-1)}{N} \\ &= \frac{p+q-1}{pq} < \frac{1}{q} + \frac{1}{p} \leq \frac{1}{2^{\lambda-1}} + \frac{1}{2^{\lambda-1}} = \frac{1}{2^{\lambda-2}} \ .\end{aligned}$$

Hence the assertion holds. □

Lemma 2. *Let M,N be two positive integers. Put $\delta = M/N - \lfloor M/N \rfloor$, hence $0 \leq \delta < 1$. Moreover, we set $U_M = U[\{0,\ldots,M-1\}]$ and $U_N = U[\{0,\ldots,N-1\}]$. Then we have*

$$\Delta(U_M \bmod N, U_N) = \frac{\delta(1-\delta)}{M/N} \leq \frac{N}{4M} \ .$$

Proof. The latter part follows from the fact that $\delta(1-\delta)$ attains the maximum value $1/4$ at $\delta = 1/2$. For the former part, we note that

$$\Pr[U_M \bmod N = a] = \begin{cases} \dfrac{\lfloor M/N \rfloor + 1}{M} > \dfrac{1}{N} & \text{for } 0 \le a \le M - \lfloor M/N \rfloor N - 1 \ , \\ \dfrac{\lfloor M/N \rfloor}{M} \le \dfrac{1}{N} & \text{for } M - \lfloor M/N \rfloor N \le a \le N - 1 \ . \end{cases}$$

This implies that

$$\begin{aligned} \Delta(U_M \bmod N, U_N) &= (M - \lfloor M/N \rfloor N) \cdot \left(\frac{\lfloor M/N \rfloor + 1}{M} - \frac{1}{N} \right) \\ &= N\delta \cdot \left(\frac{M/N - \delta + 1}{M} - \frac{1}{N} \right) = N\delta \cdot \frac{-\delta + 1}{M} = \frac{\delta(1-\delta)}{M/N} \ . \end{aligned}$$

Hence the assertion holds. □

Lemma 3. *Let L, M, N be positive integers with $L \le N$. Let $a \in \{0, \ldots, L-1\}$, and let $A = \{k \in \{0, \ldots, N-1\} \mid k \bmod L = a\}$, $K = \lfloor (N-1-a)/L \rfloor + 1$, and $\delta = M/K - \lfloor M/K \rfloor$. Moreover, we set $U_M = U[\{0, \ldots, M-1\}]$. Then we have $a + (m \bmod K) \cdot L \in A$ for any $m \in \{0, \ldots, M-1\}$, and*

$$\Delta\big(a + (U_M \bmod K) \cdot L, U[A]\big) = \frac{\delta(1-\delta)}{M/K} \le \frac{K}{4M} \ .$$

Proof. We have $a \ge 0$ and $a - L < 0$ by the choice of a. On the other hand, since $(N-1-a)/L < K \le (N-1-a)/L + 1$, we have $a + (K-1) \cdot L \le N-1$ and $a + K \cdot L > N-1$. Hence it follows that $A = \{a + k \cdot L \mid k \in \{0, \ldots, K-1\}\}$, which yields the first part of the assertion and also implies the second assertion by Lemma 2. □

For any probabilistic algorithm $\mathcal{A}$ with input x and internal randomness r, we may write $\mathcal{A}(x; r)$ instead of $\mathcal{A}(x)$ in order to emphasize the choice of the internal randomness r. We often abbreviate the term "probabilistic polynomial-time" to "PPT". In this paper, for simplifying the argument, we adopt a convention about non-uniform algorithms in a way that an advice for such an algorithm depends solely on the security parameter λ. By using an appropriate padding to the input, our convention here can be made consistent with a standard convention where an advice depends solely on the input length for the algorithm. An advice $z = z_\lambda$ for an algorithm $\mathcal{A}$ may be either made implicit in notation or indicated by writing $\mathcal{A}^{(z_\lambda)}$ or similarly.

2.2 Indistinguishability of Random Variable Families

In this paper we refer to a standard definition (mainly adopted in the area of secure multiparty computation) of the indistinguishability between two families of random variables parameterized by not only a security parameter but also some other objects. The formulation below is essentially the same as the one in Section 7.2.1.2 of Goldreich's book [7] with slight notational modifications. A main remark here is that the notion is formulated against *non-uniform* distinguishers.

Definition 1 (Indistinguishability)**.** Let $(I_\lambda)_{\lambda \ge 1}$ be a family of subsets $I_\lambda \subseteq \{0,1\}^*$ indexed by security parameter λ. Let $X = (X_{\lambda,w})_{\lambda,w}$ and $Y = (Y_{\lambda,w})_{\lambda,w}$ be families of random variables $X_{\lambda,w}$ and $Y_{\lambda,w}$ indexed by a pair of λ and $w \in I_\lambda$.

- We say that X and Y are *computationally indistinguishable* and write $X \overset{\text{comp}}{\equiv} Y$, if for any *non-uniform* PPT algorithm $\mathcal{D}$ with some advice (called a *distinguisher*), there exists a negligible function $\varepsilon(\lambda)$ satisfying, for any $\lambda \ge 1$,

$$\big|\Pr[\mathcal{D}(1^\lambda, X_{\lambda,w}) = 1] - \Pr[\mathcal{D}(1^\lambda, Y_{\lambda,w}) = 1]\big| \le \varepsilon(\lambda) \text{ for every } w \in I_\lambda.$$

- For a function $\varepsilon(\lambda)$, we say that X and Y are $\varepsilon(\lambda)$-*close* and write $X \stackrel{\varepsilon(\lambda)}{\equiv} Y$, if $\Delta(X_{\lambda,w}, Y_{\lambda,w}) \leq \varepsilon(\lambda)$ for any $\lambda \geq 1$ and any $w \in I_\lambda$. We say that X and Y are *statistically close* and write $X \stackrel{\mathrm{stat}}{\equiv} Y$, if these are $\mathsf{negl}(\lambda)$-close.

2.3 Pseudorandom Generators

As the security notion for multiparty computation in our argument in this paper is formulated by simulators against *non-uniform* distinguishers, the security for pseudorandom generators is also considered against *non-uniform* distinguishers. The definition is as follows.

Definition 2 (Pseudorandom generators)**.** Let $\ell\colon \mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$ be a function satisfying $\ell(\lambda) > \lambda$ for each $\lambda \geq 1$. We say that a deterministic polynomial-time (in λ) algorithm $\mathcal{R} = \mathcal{R}(1^\lambda, s)$ with *seed* $s \in \{0,1\}^\lambda$ is a *secure pseudorandom generator* (*PRG*) with stretch function ℓ, if

- the output of $\mathcal{R}(1^\lambda, s)$ is an $\ell(\lambda)$-bit sequence; and
- its output distribution $\mathcal{R}(1^\lambda) = \mathcal{R}(1^\lambda, U[\{0,1\}^\lambda])$ with uniformly random seed is computationally indistinguishable (against non-uniform distinguisher) from the uniform distribution $U[\{0,1\}^{\ell(\lambda)}]$ (in the sense of Definition 1).

3 Secure Two-Party Computation

Among secure multiparty computation, in this paper we focus on the simplest case of two-party computation, though our result could be extended to the case of a larger number of parties. We also focus on the semi-honest model as described in Section 3.1. In Section 3.2, we give a formalization of the situation (of main concern in this paper) where a party uses a secure PRG as the internal randomness.

3.1 Basic Notations and Terminology

Here we summarize some notations and terminology for two-party protocols in the semi-honest model. Except some notational modifications, our argument below is based on the standard security definitions (in the "view simulator" paradigm, rather than the equivalent "ideal vs. real" paradigm) described, e.g., in Section 7.2 of [7] and in [11].

Let π be a two-party protocol between parties $\mathcal{P}_1$ and $\mathcal{P}_2$. Formally, the parties are modeled as interactive PPT Turing machines that communicate with each other by following the specification of π. In this paper, we follow a popular convention that an input for $\mathcal{P}_i$ ($i \in \{1,2\}$) consists of a security parameter 1^λ common to the two parties and an "actual" input for $\mathcal{P}_i$. Unless specified otherwise, an "actual" input for $\mathcal{P}_i$ is denoted by $x_i \in \{0,1\}^*$, the internal random tape for $\mathcal{P}_i$ at an execution of π is denoted by $r_i \in \{0,1\}^*$, and the output obtained by $\mathcal{P}_i$ after an execution of π is denoted by $\mathsf{out}_i^\pi(1^\lambda, x_1, x_2; r_1, r_2)$, or by $\mathsf{out}_i^\pi(1^\lambda, \vec{x}; \vec{r})$ in short where $\vec{x} := (x_1, x_2)$ and $\vec{r} := (r_1, r_2)$. We define $\mathsf{out}^\pi(1^\lambda, \vec{x}; \vec{r})$ to be the pair of $\mathsf{out}_1^\pi(1^\lambda, \vec{x}; \vec{r})$ and $\mathsf{out}_2^\pi(1^\lambda, \vec{x}; \vec{r})$ in this order. As the parties are PPT, we may assume that the lengths of the inputs x_1, x_2 and of the random tapes r_1, r_2 are bounded by a polynomial in λ. Moreover, for simplifying the argument, we also assume that $r_1 \in \{0,1\}^{\rho_1(\lambda)}$ and $r_2 \in \{0,1\}^{\rho_2(\lambda)}$ for some positive-valued polynomials ρ_1, ρ_2. Let X_λ denote the set of the input pairs (x_1, x_2) associated to security parameter λ.

We let the *transcript* for Party $\mathcal{P}_i$ in an execution of π mean the sequence $(m_1^{(i)}, \ldots, m_{\ell_i}^{(i)})$ of messages (in the chronological order) sent to $\mathcal{P}_i$ from the other party $\mathcal{P}_{3-i}$ during the protocol execution. Let $\mathsf{trans}_i^\pi(1^\lambda, x_1, x_2; r_1, r_2)$ or $\mathsf{trans}_i^\pi(1^\lambda, \vec{x}; \vec{r})$ denote the transcript for $\mathcal{P}_i$ in the execution of π with inputs x_1, x_2 and random tapes r_1, r_2. The *view* for $\mathcal{P}_i$ consists of x_i, r_i and $\mathsf{trans}_i^\pi(1^\lambda, \vec{x}; \vec{r})$; in particular, the view for a party involves the content of the party's random tape, which is an important fact in our argument below. Then the party $\mathcal{P}_i$ finally computes $\mathsf{out}_i^\pi(1^\lambda, \vec{x}; \vec{r})$ from the view for $\mathcal{P}_i$.

We let a *functionality* with input set $I \subseteq \{0,1\}^* \times \{0,1\}^*$ mean any pair $f = (f_1, f_2)$ of possibly probabilistic functions $f_1 = f_1(\vec{x})$ and $f_2 = f_2(\vec{x})$ with $\vec{x} = (x_1, x_2) \in I$. More precisely, for each $\vec{x} \in I$,

$f_1(\vec{x})$ and $f_2(\vec{x})$ are (possibly correlated) random variables endowed with their own internal randomness. We write $f(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$. In the following argument, unless specified otherwise, we assume that f is a functionality with input set $I = \bigcup_{\lambda \geq 1} X_\lambda$ where X_λ is the input set for a two-party protocol π with security parameter λ.

We describe the definition of security for a two-party protocol π in the semi-honest model[1]. Intuitively, the condition below means that the view for any party in an execution of π can be efficiently recovered, in a computationally indistinguishable manner, solely from the local input and the local output for the party. Note that the following definition implies also that the protocol computes the value of f correctly.

Definition 3 (Security in the semi-honest model)**.** Let π be a two-party protocol to compute a functionality $f = (f_1, f_2)$. For $i \in \{1, 2\}$, we say that π is *secure against semi-honest Party* $\mathcal{P}_i$ with *computational* (respectively, *statistical*) *simulator*, if there exists a PPT algorithm $\mathcal{S}_i$, called a *simulator* for $\mathcal{P}_i$, satisfying that the two probability distributions

$$\left(\mathcal{S}_i(1^\lambda, x_i, f_i(\vec{x})), f(\vec{x})\right)_{\lambda, \vec{x}}$$

and

$$\left(\left[(x_i, r_i, \mathsf{trans}_i^\pi(1^\lambda, \vec{x}; \vec{r}), \mathsf{out}^\pi(1^\lambda, \vec{x}; \vec{r})) \colon r_1 \leftarrow\!\!\!\!\!\!\rightharpoonup U[\{0,1\}^{\rho_1(\lambda)}], r_2 \leftarrow\!\!\!\!\!\!\rightharpoonup U[\{0,1\}^{\rho_2(\lambda)}]\right]\right)_{\lambda, \vec{x}}$$

are computationally indistinguishable (respectively, statistically close), where the indices λ and $\vec{x}$ run over the ranges $\lambda \geq 1$ and $\vec{x} \in X_\lambda$.

3.2 The Case of Using a PRG

From now, we consider an extension of the definition above to the case where one of the two parties uses a secure PRG for generating the party's random tape. Recall that, a motivation of including a party's random tape to the party's view (hence to a simulator's output as well) in the standard security formulation comes from an observation that a corrupted (semi-honest) party in a practical situation might be able to see the random tape which is stored in the party's own device, therefore any extra information should not be yielded by the random tape used in a protocol execution. Accordingly, when such a party uses a PRG, it is reasonable that the seed for the PRG is also included to the party's view (and also to a simulator's output).

The following formulation is based on the observation above. Let π be a two-party protocol as in Section 3.1. Let $i \in \{1, 2\}$, and let $\mathcal{R}$ be a secure PRG with stretch function $\ell(\lambda) = \rho_i(\lambda)$. Then we regard "the execution of π where Party $\mathcal{P}_i$ uses a PRG $\mathcal{R}$" as the following two-party protocol, denoted by $\pi \circ_i \mathcal{R}$:

- The input set for $\pi \circ_i \mathcal{R}$ is the same as π, and the sets of random tapes of Party $\mathcal{P}_i$ and Party $\mathcal{P}_{3-i}$ for $\pi \circ_i \mathcal{R}$ are $\{0,1\}^\lambda$ and $\{0,1\}^{\rho_{3-i}(\lambda)}$, respectively.
- Given a security parameter 1^λ, local inputs x_1, x_2 for two parties, and random tapes $s_i \leftarrow\!\!\!\!\!\!\rightharpoonup U[\{0,1\}^\lambda]$ and $r_{3-i} \leftarrow\!\!\!\!\!\!\rightharpoonup U[\{0,1\}^{\rho_{3-i}(\lambda)}]$ for Parties $\mathcal{P}_i$ and $\mathcal{P}_{3-i}$, respectively, to execute the protocol $\pi \circ_i \mathcal{R}$, first $\mathcal{P}_i$ runs $\mathcal{R}(1^\lambda, s_i)$ with seed s_i and obtains its output $r_i \in \{0,1\}^{\rho_i(\lambda)}$. Then the two parties $\mathcal{P}_1, \mathcal{P}_2$ jointly execute the protocol π with security parameter λ, input pair (x_1, x_2), and random tapes r_1, r_2.

Here we emphasize that, the view for the party $\mathcal{P}_i$ in the new protocol $\pi \circ_i \mathcal{R}$ involves the seed s_i for the PRG $\mathcal{R}$ rather than the random tape r_i of $\mathcal{P}_i$ for the original protocol π. Accordingly, a simulator to prove the security of $\pi \circ_i \mathcal{R}$ against $\mathcal{P}_i$ has to simulate the seed s_i as well as the other part of the party's view. On the other hand, we note that the pseudorandom tape r_i used during the protocol execution can be deterministically recovered from s_i, which is included in the view for $\mathcal{P}_i$ in the new protocol $\pi \circ_i \mathcal{R}$.

[1]The notion is often called with different names in the literature; e.g., "π privately computes f" in Section 7.2 of [7]; and "π securely computes f in the presence of static semi-honest adversaries" in [11].

4 Security When Using a PRG: First Negative Results

By a cryptographer's daily-life intuition, it is expected that computational security of a cryptographic protocol should be preserved when its internal true randomness is replaced by an output of a secure PRG. Nevertheless, as opposed to this expectation, in this section we give (under a certain computational assumption) a concrete example of a situation where a secure two-party protocol becomes insecure when a party uses a secure PRG to generate its internal random tape.

In order to state the main result of this section, we prepare some terminology. Recall that an integer N is called a *Blum integer* if it is of the form $N = pq$ with p, q being distinct primes congruent to 3 modulo 4. We say that a deterministic algorithm $\mathcal{B} = \mathcal{B}(1^\lambda)$ is a *Blum integer generator*, if its output $\mathcal{B}(1^\lambda)$ (for $\lambda \geq 5$)[2] is a Blum integer with two prime factors having λ-bit lengths. We say that a Blum integer generator $\mathcal{B}$ is *efficiently factorizable*, if there is a PPT *uniform*[3] algorithm $\mathcal{F}$ satisfying that $\mathcal{F}(\mathcal{B}(1^\lambda))$ is a prime factor of $\mathcal{B}(1^\lambda)$ with probability $\Omega(1)$. Then our result is stated as follows:

Theorem 1. *Assume that there exists a polynomial-time Blum integer generator that is not efficiently factorizable (see above for the terminology). Assume moreover that a secure PRG exists. Then there exist a two-party protocol π and a* secure *PRG $\mathcal{R}$ with the following two properties:*

- *π is secure against semi-honest Party $\mathcal{P}_1$ (with statistical simulator).*
- *$\pi \circ_1 \mathcal{R}$ is* not *secure against semi-honest Party $\mathcal{P}_1$ (even with computational simulator).*

Section 4.1 is devoted to the preliminaries towards constructing the two-party protocol and the PRG in Theorem 1. Section 4.2 gives the two-party protocol and its security proof, which is the first part of the claim in Theorem 1. The construction of the PRG is given in Section 4.3, where we also give a proof of the second part of the claim in Theorem 1 and hence conclude the proof of the theorem.

4.1 Preliminaries: The Rabin Function

Here we summarize some facts about the Rabin function [14] used in our argument below. The Rabin function modulo N computes $x^2 \bmod N$ for a given integer $x \in (\mathbb{Z}/N\mathbb{Z})^\times$, where $N = pq$ is the product of two distinct primes p, q of the same bit length with $p \equiv q \equiv 3 \pmod 4$. It is known [14] that factoring the composite N is polynomial-time reducible to inverting Rabin function modulo the N and vice versa. Let

$$\mathsf{QR}_N \stackrel{\text{def}}{=} \{x^2 \bmod N \mid x \in (\mathbb{Z}/N\mathbb{Z})^\times\}$$

be the set of quadratic residues modulo N, which is by definition equal to the image of Rabin function modulo N. Each $y \in \mathsf{QR}_N$ has four preimages for the function (i.e., square roots modulo N). Namely, we have a decomposition $(\mathbb{Z}/N\mathbb{Z})^\times \simeq (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ owing to Chinese Remainder Theorem. Then, for $y = x^2$ with $x \in (\mathbb{Z}/N\mathbb{Z})^\times$, the four pairs $(\pm x \bmod p, \pm x \bmod q)$ with two choices of each sign represent the square roots of y modulo N.

We recall the following fact for finding a square root modulo a composite $N = pq$ as in the Rabin function when a prime factor of N is known:

Lemma 4. *There exists a PPT algorithm, with the N, p (or q) and some $y \in (\mathbb{Z}/N\mathbb{Z})^\times$ as inputs, that outputs an element x of $(\mathbb{Z}/N\mathbb{Z})^\times$ satisfying that, if $y \in \mathsf{QR}_N$, then x is uniformly random among the four square roots of y modulo N.*

Proof. As $p \equiv q \equiv 3 \pmod 4$, both $p' = (p+1)/4$ and $q' = (q+1)/4$ are integers. The algorithm runs in the following four steps: (i) Compute $y_p \leftarrow y \bmod p$, $y_q \leftarrow y \bmod q$, $z_p \leftarrow {y_p}^{p'}$, and $z_q \leftarrow {y_q}^{q'}$. (ii) Choose $x_p \leftarrow\!\!\shortmid U[\{z_p, -z_p\}]$ and $x_q \leftarrow\!\!\shortmid U[\{z_q, -z_q\}]$. (iii) Compute the unique element $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ corresponding to the pair $(x_p, x_q) \in (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$. (iv) Output the x.

[2] Note that, for $\lambda \leq 4$, there is at most one λ-bit prime congruent to 3 modulo 4.

[3] We note that a non-uniform algorithm with advice can trivially factorize the deterministic output of $\mathcal{B}(1^\lambda)$.

The whole computation can be done in polynomial time with respect to the bit length of N, since a factor of N is known. From now, we suppose $y \in \mathsf{QR}_N$, therefore $y = w^2$ for some $w \in (\mathbb{Z}/N\mathbb{Z})^\times$. Put $w_p = w \bmod p$ and $w_q = w \bmod q$. Then we have $y_p = {w_p}^2$ and ${z_p}^2 = {y_p}^{2p'} = {w_p}^{4p'} = {w_p}^{p+1}$, which is equal (in $\mathbb{Z}/p\mathbb{Z}$) to ${w_p}^2 = y_p$ by Fermat's Little Theorem. Hence we have $(\pm z_p)^2 = y_p$, and we have $(\pm z_q)^2 = y_q$ similarly. This implies that the elements of $(\mathbb{Z}/N\mathbb{Z})^\times$ corresponding to $(\pm z_p, \pm z_q)$ are the four square roots of y. This completes the proof. □

On the other hand, although it is (believed to be) computationally hard to find a square root modulo the N above of a *given* quadratic residue, the next lemma shows that (approximately) uniform sampling of a pair (x, y) of a *random* quadratic residue y and its square root x is still computationally feasible with high probability. Precisely, let $N = pq$ be as above and let λ be the common bit length of p and q. We consider the following algorithm, which is given 1^λ and N as inputs but *not* given any prime factors p, q of N:

1. Repeat the following process up to λ times until an appropriate $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ is found:
   - Compute $a \leftarrow r \bmod N$ with $r \leftarrowtail U[\{0,1\}^{3\lambda}]$, and check if $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ and $\left(\frac{a}{N}\right) = -1$ where $\left(\frac{a}{N}\right)$ denotes the Jacobi symbol of a modulo N.

   In case where such an a has not been found, output a pair $(1 \bmod N, -1 \bmod N)$ and stop.
2. Compute $x \leftarrow r \bmod N$ with $r \leftarrowtail U[\{0,1\}^{3\lambda}]$, and if $x \notin (\mathbb{Z}/N\mathbb{Z})^\times$, then output a pair $(1 \bmod N, -1 \bmod N)$ and stop.
3. Choose y from the four elements $\pm x^2 \bmod N$ and $\pm ax^2 \bmod N$ uniformly at random, by using two random bits. Then output (x, y).

Note that the output (x, y) of this algorithm always satisfies $x, y \in (\mathbb{Z}/N\mathbb{Z})^\times$. Note also that the complexity of the algorithm is polynomial in λ; indeed, the Jacobi symbol $\left(\frac{a}{N}\right)$ can be computed without knowledge of prime factors of N by using Law of Quadratic Reciprocity. Now the following property holds.

Lemma 5. *The output (x, y) of the algorithm above satisfies the following:*

- *The distribution of y is statistically close to $U[(\mathbb{Z}/N\mathbb{Z})^\times]$, where the bound of the statistical distance is dependent solely on λ.*
- *If $y \in \mathsf{QR}_N$, then the conditional distribution of x conditioned on the y is statistically close to uniform over the four square roots of y, where the bound is again dependent solely on λ.*

Proof. First, we analyze Step 1. For each of the repeated processes, the combination of Lemmas 1 and 2 as well as the fact $N \leq 2^{2\lambda}$ implies that, the statistical distance between $U[(\mathbb{Z}/N\mathbb{Z})^\times]$ and the distribution of the element a is at most $N/2^{3\lambda+2} + 2^{-(\lambda-2)} \leq 2^{-(\lambda+2)} + 2^{-(\lambda-2)} < 2^{-(\lambda-1)}$. On the other hand, for $a' \leftarrowtail U[(\mathbb{Z}/N\mathbb{Z})^\times]$, we have $\left(\frac{a'}{N}\right) = 1$ with probability $1/2$. This implies that, the a satisfies either $a \notin (\mathbb{Z}/N\mathbb{Z})^\times$ or $\left(\frac{a}{N}\right) = 1$ with probability at most $1/2 + 2^{-(\lambda-1)}$. Therefore, the probability, denoted by ρ_1, that the algorithm stops at Step 1 is at most $\rho_1' \stackrel{\text{def}}{=} (1/2 + 2^{-(\lambda-1)})^\lambda$, the latter being negligible in λ and dependent solely on λ.

Secondly, we analyze Step 2. By the choice of x, each element of $(\mathbb{Z}/N\mathbb{Z})^\times$ appears as the value of x with probability $\lfloor 2^{3\lambda}/N \rfloor / 2^{3\lambda}$ or $(\lfloor 2^{3\lambda}/N \rfloor + 1)/2^{3\lambda}$. On the other hand, by Lemmas 1 and 2 and the fact $N \leq 2^{2\lambda}$ again, the probability, denoted by ρ_2, that $x \notin (\mathbb{Z}/N\mathbb{Z})^\times$ is at most $2^{-(\lambda+2)} + 2^{-(\lambda-2)} < \rho_2' \stackrel{\text{def}}{=} 2^{-(\lambda-1)}$, the latter being negligible in λ and dependent solely on λ. Hence, regarding Steps 1 and 2, for each element of $(\mathbb{Z}/N\mathbb{Z})^\times$, the probability that the algorithm has not stopped at Step 1 and this element appears as the value of x at Step 2 is either α or $\alpha + \delta$, where $\alpha \stackrel{\text{def}}{=} (1 - \rho_1)\lfloor 2^{3\lambda}/N \rfloor / 2^{3\lambda}$ and $\delta \stackrel{\text{def}}{=} (1 - \rho_1)/2^{3\lambda}$. On the other hand, the algorithm stops before arriving at Step 3 with probability $\rho_1 + (1 - \rho_1)\rho_2 \leq \rho_1' + \rho_2'$, the latter being negligible in λ and dependent solely on λ.

Thirdly, we analyze Step 3. First we show that, $x^2 \bmod N$ is the only choice among the four candidates of y for being a quadratic residue. To see this, recall that $\left(\frac{a}{N}\right) = -1$, therefore precisely one of $a \bmod p$ and $a \bmod q$ is a quadratic residue modulo p and q, respectively. Say, $a \bmod p$ is a quadratic residue and $a \bmod q$ is not. Note also that, since $p \equiv q \equiv 3 \pmod 4$, neither $-1 \bmod p$ nor $-1 \bmod q$ is a quadratic residue. Now none of $-x^2 \bmod p$, $ax^2 \bmod q$, and $-ax^2 \bmod p$ is a quadratic residue, which implies that none of $-x^2 \bmod N$ and $\pm ax^2 \bmod N$ is a quadratic residue, too. Hence the claim of this paragraph holds.

By the previous paragraph, an element $y \in \mathsf{QR}_N$ is chosen at Step 3 if and only if one of the four square roots of y is chosen at Step 2 and then $x^2 \bmod N$ is chosen at Step 3 (with probability 1/4). Hence, the probability, denoted by P_y, that the y is chosen satisfies

$$4\alpha \cdot \frac{1}{4} = \alpha \leq P_y \leq 4(\alpha + \delta) \cdot \frac{1}{4} = \alpha + \delta \ .$$

On the other hand, for each square root x of y, the probability, denoted by $Q_{x,y}$, that the pair (x, y) is chosen satisfies

$$\alpha \cdot \frac{1}{4} = \frac{\alpha}{4} \leq Q_{x,y} \leq (\alpha + \delta) \cdot \frac{1}{4} = \frac{\alpha + \delta}{4} \ .$$

Therefore, the conditional probability of the choice of x conditioned on the choice of y satisfies

$$\frac{\alpha}{4} \cdot \frac{1}{\alpha + \delta} = \frac{\alpha}{4(\alpha + \delta)} \leq \frac{Q_{x,y}}{P_y} \leq \frac{\alpha + \delta}{4} \cdot \frac{1}{\alpha} = \frac{\alpha + \delta}{4\alpha} \ .$$

The differences of the upper and lower bounds for $Q_{x,y}/P_y$ from the probability 1/4 of the uniformly random choice are evaluated as

$$\frac{\alpha + \delta}{4\alpha} - \frac{1}{4} = \frac{\delta}{4\alpha} \leq \frac{2^{-3\lambda}}{4(1 - \rho_1')(1 - 2^{-\lambda}) \cdot 2^{-2\lambda}} = \frac{1}{4(1 - \rho_1')(1 - 2^{-\lambda}) \cdot 2^{\lambda}}$$

and

$$\frac{1}{4} - \frac{\alpha}{4(\alpha + \delta)} = \frac{\delta}{4(\alpha + \delta)} \leq \frac{\delta}{4\alpha}$$

where we used the relations $\delta \leq 2^{-3\lambda}$ and

$$\alpha \geq (1 - \rho_1) \left(\frac{2^{3\lambda}}{N} - 1 \right) \cdot \frac{1}{2^{3\lambda}} \geq (1 - \rho_1) \left(\frac{2^{3\lambda}}{2^{2\lambda}} - 1 \right) \cdot \frac{1}{2^{3\lambda}} \geq (1 - \rho_1') \left(1 - \frac{1}{2^{\lambda}} \right) \cdot \frac{1}{2^{2\lambda}} \ .$$

Hence we have

$$\left| \frac{Q_{x,y}}{P_y} - \frac{1}{4} \right| \leq \frac{1}{4(1 - \rho_1')(1 - 2^{-\lambda}) \cdot 2^{\lambda}} \ ,$$

which is negligible in λ and is dependent solely on λ, since ρ_1' has the same property. This implies the second assertion of this lemma.

Finally, for the first assertion of this lemma, owing to the argument above, we may assume without loss of generality (except only negligible differences dependent solely on λ) that the algorithm has not stopped before Step 3 and the element x chosen in Step 2 is uniformly random over $(\mathbb{Z}/N\mathbb{Z})^{\times}$. It follows that $x^2 \bmod N$ is uniformly random over QR_N. Now by symmetry, we may assume without loss of generality (as we already did above) that $a \bmod p$ is a quadratic residue modulo p and $a \bmod q$ is not a quadratic residue modulo q. This implies that ± 1 and $\pm a$ are the representatives of the four cosets for the subgroup QR_N in $(\mathbb{Z}/N\mathbb{Z})^{\times}$; in fact, $\left(\left(\frac{z}{p}\right), \left(\frac{z}{q}\right)\right)$ is equal to $(1, 1)$ for $z = 1$; $(1, -1)$ for $z = a$; $(-1, 1)$ for $z = -a$; and $(-1, -1)$ for $z = -1$. Since x^2 is uniformly random over QR_N as mentioned above, it follows that the choice of y is uniformly random over $(\mathbb{Z}/N\mathbb{Z})^{\times}$. This completes the proof of Lemma 5. □

(Party $\mathcal{P}_1$)	Input: $N = pq$ (p, q are unknown for $\mathcal{P}_1$) Random tape: $r_1 \in \{0,1\}^{3\lambda}$	Output: (none)
(Party $\mathcal{P}_2$)	Input: λ-bit primes $p \neq q$ with $p \equiv q \equiv 4 \pmod 3$ Random tape: $r_2 \in \{0,1\}^2$	Output: (none)

1. $\mathcal{P}_1$ computes $y \leftarrow r_1 \bmod N$ (where r_1 is regarded as a binary representation of an integer), and sends N and y to $\mathcal{P}_2$.
2. $\mathcal{P}_2$ decides if $y \in \mathsf{QR}_N$ or not, based on Chinese Remainder Theorem and Law of Quadratic Reciprocity by using p and q.
3. – If $y \in \mathsf{QR}_N$, then $\mathcal{P}_2$ computes a uniformly random square root x of y modulo N by using the two random bits in r_2 as in Lemma 4 of Section 4.1, and sends x to $\mathcal{P}_1$.
   – If $y \notin \mathsf{QR}_N$, then $\mathcal{P}_2$ sends $\perp$ to $\mathcal{P}_1$.

Figure 1: The two-party protocol for Theorem 1

4.2 The Protocol

Here we construct the two-party protocol π in the statement of Theorem 1. The protocol π is described in Figure 1. We note that the security of the protocol against $\mathcal{P}_2$ is trivial (as $\mathcal{P}_2$ receives the input N of $\mathcal{P}_1$) though our argument concerns the security against $\mathcal{P}_1$ only. We also note that the protocol π has no output, therefore the simulator constructed in the security proof may ignore the part of its input corresponding to the empty output of π. Now we have the following result.

Proposition 1. *The protocol π is secure against semi-honest $\mathcal{P}_1$ with statistical simulator.*

Proof. We construct a simulator $\mathcal{S}_1$ for $\mathcal{P}_1$ with input 1^λ and N. First we focus on the computation $y = r_1 \bmod N$ in Step 1. In a real execution of π, the distribution of r_1 conditioned on a chosen y is uniform over the set $\{k \in \{0, \ldots, 2^{3\lambda} - 1\} \mid k \bmod N = y\}$. Now Lemma 3 implies that the output distribution of a probabilistic function $g(y) = y + (u \bmod K_y) \cdot N$ with $u \leftarrow\!\!\!\circ\, U[\{0,1\}^{4\lambda}]$, where $K_y = \lfloor (2^{3\lambda} - 1 - y)/N \rfloor + 1$, is statistically close to the conditional distribution of the r_1 (with bound dependent solely on λ).

On the other hand, in a real execution of π, the element y chosen in Step 1 is statistically close to $U[\mathbb{Z}/N\mathbb{Z}] \stackrel{\mathrm{stat}}{\equiv} U[(\mathbb{Z}/N\mathbb{Z})^\times]$ (with bound dependent solely on λ) owing to Lemmas 1 and 2. Moreover, by Lemma 4, the message received by $\mathcal{P}_1$ at Step 3, denoted here by η, in the real execution of π is a uniformly random square root x of y in $\mathbb{Z}/N\mathbb{Z}$ if $y \in \mathsf{QR}_N$, and it is always $\perp$ if $y \notin \mathsf{QR}_N$. Now let (x', y') denote an output of the algorithm in Lemma 5 (recall that this algorithm does not use knowledge of prime factors of N), and let η' denote an element computed in the same way as η but by using (x', y') instead of (x, y). Then by Lemma 5, we have $(x', y') \stackrel{\mathrm{stat}}{\equiv} (x, y)$, therefore $(r_1, x, y, \eta) \stackrel{\mathrm{stat}}{\equiv} (g(y), x, y, \eta) \stackrel{\mathrm{stat}}{\equiv} (g(y'), x', y', \eta')$. According to these arguments, the simulator $\mathcal{S}_1$ can output (within polynomial time) $g(y')$ as the simulated random tape for $\mathcal{P}_1$ and η' as the simulated transcript at Step 3, and the simulation is statistically close to the real (i.e., $(g(y'), \eta') \stackrel{\mathrm{stat}}{\equiv} (r_1, \eta)$) by the argument above. This completes the proof. □

4.3 The PRG

Here we construct the secure PRG $\mathcal{R}$ in the statement of Theorem 1. First, by the hypothesis of Theorem 1, there exists a polynomial-time Blum integer generator $\mathcal{B}$ that is not efficiently factorizable. Recall that $\mathcal{B}$ is a deterministic algorithm by definition. Secondly, there exists a secure PRG by the hypothesis of Theorem 1 again. As the simulator $\mathcal{S}_1$ constructed in the proof of Proposition 1 is PPT, a standard technique to securely expand the output length of a PRG (see e.g., Section 3.3 of [6]) yields a secure PRG $\mathcal{R}_0$ with stretch function ℓ_0 satisfying that $\ell_0(\lambda)$ equals the length of the internal random tape for $\mathcal{S}_1$ with security parameter λ. Now we construct the PRG $\mathcal{R}$ with stretch function $\ell(\lambda) = 3\lambda$ as follows:

1. Given security parameter 1^λ and a random seed $s \leftarrow\!\!\!\circ\, U[\{0,1\}^\lambda]$, the algorithm first execute $\mathcal{R}_0(1^\lambda, s)$ and obtain its output $\widetilde{r} \in \{0,1\}^{\ell_0(\lambda)}$.

2. Secondly, the algorithm executes $\mathcal{S}_1(1^\lambda, \mathcal{B}(1^\lambda); \widetilde{r})$ and obtains its output $(N, r_1, \mathsf{trans}_1)$. Then the algorithm outputs $r_1 \in \{0,1\}^{3\lambda}$.

Proposition 2. *The PRG $\mathcal{R}$ is secure.*

Proof. Assume, for the contrary, that a PPT distinguisher $\mathcal{D}$ satisfies that $P_1 \stackrel{\text{def}}{=} \Pr_{r_1 \leftarrow \mathcal{R}(1^\lambda)}[\mathcal{D}(1^\lambda, r_1) = 1]$ and $P_2 \stackrel{\text{def}}{=} \Pr_{r_1 \leftarrow\!\!\!\!\circ\, U[\{0,1\}^{3\lambda}]}[\mathcal{D}(1^\lambda, r_1) = 1]$ have a non-negligible difference. First note that, as the simulator $\mathcal{S}_1$ for $\mathcal{P}_1$ in the protocol π is statistically close to the real protocol execution by Proposition 1, it follows that the distribution of r_1 generated by $(N, r_1, \mathsf{trans}_1) \leftarrow \mathcal{S}_1(1^\lambda, \mathcal{B}(1^\lambda); \widetilde{r})$ with $\widetilde{r} \leftarrow\!\!\!\!\circ\, U[\{0,1\}^{\ell_0(\lambda)}]$ is statistically close to $U[\{0,1\}^{3\lambda}]$. This implies that $P_3 \stackrel{\text{def}}{=} \Pr_{(N, r_1, \mathsf{trans}_1) \leftarrow \mathcal{S}_1(1^\lambda, \mathcal{B}(1^\lambda); U[\{0,1\}^{\ell_0(\lambda)}])}[\mathcal{D}(1^\lambda, r_1) = 1]$ has a negligible difference from P_2, therefore P_1 and P_3 should have a non-negligible difference. Now we consider the following distinguisher $\overline{\mathcal{D}}$ for $\mathcal{R}_0$;

- given 1^λ and $\widetilde{r} \in \{0,1\}^{\ell_0(\lambda)}$, the distinguisher computes $(N, r_1, \mathsf{trans}_1) \leftarrow \mathcal{S}_1(1^\lambda, \mathcal{B}(1^\lambda); \widetilde{r})$ and outputs the output value of $\mathcal{D}(1^\lambda, r_1)$.

Then we have $P_3 = \Pr_{\widetilde{r} \leftarrow\!\!\!\!\circ\, U[\{0,1\}^{\ell_0(\lambda)}]}[\overline{\mathcal{D}}(1^\lambda, \widetilde{r}) = 1]$, while $P_1 = \Pr_{\widetilde{r} \leftarrow\!\!\!\!\circ\, \mathcal{R}_0(1^\lambda)}[\overline{\mathcal{D}}(1^\lambda, \widetilde{r}) = 1]$ by the construction of $\mathcal{R}$. As $\overline{\mathcal{D}}$ is PPT as well as $\mathcal{D}$, the non-negligible difference of P_1 and P_3 mentioned above contradicts the security of $\mathcal{R}_0$. Hence $\mathcal{R}$ is secure, which completes the proof. □

To complete the proof of Theorem 1, we give the following result.

Proposition 3. *If there exists a PPT computational simulator $\widetilde{\mathcal{S}}_1$ for Party $\mathcal{P}_1$ in the protocol $\pi \circ_1 \mathcal{R}$, then there exists a PPT uniform algorithm $\mathcal{F}$ that outputs a prime factor of the Blum integer $\mathcal{B}(1^\lambda)$ with probability $\Omega(1)$.*

Proof. Before constructing the algorithm $\mathcal{F}$ in the statement, first we define the following auxiliary algorithm $\mathcal{F}_0$, where its input consists of 1^λ, an integer N of at most (2λ)-bit length, a bit sequence $\widetilde{r}_1 \in \{0,1\}^{\ell_0(\lambda)}$, and an object $\widetilde{x}$:

1. The algorithm executes $\mathcal{S}_1(1^\lambda, N; \widetilde{r}_1)$ and obtains a triple (N, r_1'', x''). Now the algorithm aborts unless $\widetilde{x} \in (\mathbb{Z}/N\mathbb{Z})^\times$, $x'' \in (\mathbb{Z}/N\mathbb{Z})^\times$, and $x'' \notin \{\widetilde{x}, -\widetilde{x}\}$.
2. The algorithm outputs $\gcd(\widetilde{x} - x'', N)$.

Note that $\mathcal{F}_0$ is a PPT uniform algorithm. By using this, we construct the algorithm $\mathcal{F}$ as follows:

1. Given 1^λ and $N_\lambda \stackrel{\text{def}}{=} \mathcal{B}(1^\lambda)$ as inputs, the algorithm executes $\widetilde{\mathcal{S}}_1(1^\lambda, N_\lambda)$ and obtains a triple $(N_\lambda, \overline{r}_1, \overline{x})$ where either $\overline{x} \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$ or $\overline{x} = \perp$.
2. The algorithm executes $\mathcal{R}_0(1^\lambda, \overline{r}_1)$ and obtains $\widetilde{r}_1 \in \{0,1\}^{\ell_0(\lambda)}$.
3. The algorithm executes $\mathcal{F}_0(1^\lambda, N_\lambda, \widetilde{r}_1, \overline{x})$; if this $\mathcal{F}_0$ aborts then the algorithm also aborts, otherwise the algorithm outputs the output value of the $\mathcal{F}_0$.

Note that $\mathcal{F}$ is a PPT uniform algorithm as well as $\mathcal{F}_0$.

Now we consider the following distinguisher $\mathcal{D}_1$ for $(N, \overline{r}_1, \overline{x})$ which is either a real view for $\mathcal{P}_1$ in $\pi \circ_1 \mathcal{R}$ or an output of the simulator $\widetilde{\mathcal{S}}_1(1^\lambda, N)$:

1. The distinguisher executes $\mathcal{R}_0(1^\lambda, \overline{r}_1)$ and obtains $\widetilde{r}_1 \in \{0,1\}^{\ell_0(\lambda)}$.
2. The distinguisher executes $\mathcal{F}_0(1^\lambda, N, \widetilde{r}_1, \overline{x})$; if this $\mathcal{F}_0$ aborts then the distinguisher outputs 0, otherwise we denote the output value of the $\mathcal{F}_0$ by p.
3. The distinguisher outputs 1 if p is a non-trivial divisor of N, otherwise outputs 0.

Note that $\mathcal{D}_1$ is PPT. Now by the construction of the algorithms, the probability, denoted by $P_{\mathcal{F}}$, that $\mathcal{F}$ succeeds to factorize the given integer N_λ is equal to

$$P_{\mathcal{F}} = \Pr_{(N_\lambda, \overline{r}_1, \overline{x}) \leftarrow \widetilde{\mathcal{S}}_1(1^\lambda, N_\lambda)} [\mathcal{D}_1(1^\lambda, N_\lambda, \overline{r}_1, \overline{x}) = 1] \ .$$

Now let p_λ and q_λ be the two prime factors of N_λ. As $\widetilde{\mathcal{S}}_1$ is computationally indistinguishable from the real execution of $\pi \circ_1 \mathcal{R}$ by the hypothesis, it follows that $|P_{\mathcal{F}} - P_1|$ is negligible, where

$$P_1 \stackrel{\text{def}}{=} \Pr_{\overline{r}_1 \leftarrow\!\!\$\, U[\{0,1\}^\lambda], \overline{x} \leftarrow \mathsf{trans}_1^{\pi \circ_1 \mathcal{R}}(1^\lambda, N_\lambda, (p_\lambda, q_\lambda); \overline{r}_1, U[\{0,1\}^2])} [\mathcal{D}_1(1^\lambda, N_\lambda, \overline{r}_1, \overline{x}) = 1] \ .$$

Secondly, we consider the following *non-uniform* distinguisher $\mathcal{D}_2$ for PRG $\mathcal{R}_0$, where the advice for $\mathcal{D}_2$ associated to the security parameter λ consists of the p_λ, q_λ, and N_λ:

1. Given 1^λ and $\widetilde{r}_1 \in \{0,1\}^{\ell_0(\lambda)}$ as input and p_λ, q_λ, and N_λ as advice, the distinguisher executes $\mathcal{S}_1(1^\lambda, N_\lambda; \widetilde{r}_1)$ and obtains a triple (N_λ, r_1'', x'').
2. The distinguisher emulates the execution of π with input N_λ and random tape r_1'' for $\mathcal{P}_1$ and input (p_λ, q_λ) and a uniformly random tape r_2 for $\mathcal{P}_2$. Let $\overline{x}$ denote the transcript received by $\mathcal{P}_1$ during the emulated execution of π.
3. The distinguisher outputs 0 unless $\overline{x} \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$, $x'' \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$, and $x'' \notin \{\overline{x}, -\overline{x}\}$.
4. The distinguisher computes $p \stackrel{\text{def}}{=} \gcd(\overline{x} - x'', N_\lambda)$, and outputs 1 if $p \in \{p_\lambda, q_\lambda\}$, otherwise outputs 0.

The constructions of algorithms $\mathcal{D}_1$ and $\mathcal{F}_0$ imply that, when $\lambda \geq 5$, the probability P_1 above can be written as

$$P_1 = \Pr_{\widetilde{r}_1 \leftarrow\!\!\$\, \mathcal{R}_0(1^\lambda)} [\mathcal{D}_2^{(p_\lambda, q_\lambda, N_\lambda)}(1^\lambda, \widetilde{r}_1) = 1] \ .$$

As the PRG $\mathcal{R}_0$ is secure *against non-uniform distinguishers* (due to our security definition for PRGs) and $\mathcal{D}_2$ is PPT, it follows that $|P_1 - P_2|$ is negligible, where

$$P_2 \stackrel{\text{def}}{=} \Pr_{\widetilde{r}_1 \leftarrow\!\!\$\, U[\{0,1\}^{\ell_0(\lambda)}]} [\mathcal{D}_2^{(p_\lambda, q_\lambda, N_\lambda)}(1^\lambda, \widetilde{r}_1) = 1] \ .$$

Thirdly, we consider the following *non-uniform* distinguisher $\mathcal{D}_3$ for simulator $\mathcal{S}_1$ for Party $\mathcal{P}_1$ in π, where the advice for $\mathcal{D}_3$ associated to the security parameter λ consists of the p_λ, q_λ, and N_λ:

1. Given 1^λ and a triple (N, r_1', x') as input and p_λ, q_λ, and N_λ as advice, the distinguisher outputs 0 if $N \neq N_\lambda$.
2. The distinguisher emulates the execution of π with input N_λ and random tape r_1' for $\mathcal{P}_1$ and input (p_λ, q_λ) and a uniformly random tape r_2 for $\mathcal{P}_2$. Let $\overline{x}$ denote the transcript received by $\mathcal{P}_1$ during the emulated execution of π.
3. The distinguisher outputs 0 unless $\overline{x} \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$, $x' \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$, and $x' \notin \{\overline{x}, -\overline{x}\}$.
4. The distinguisher computes $p \stackrel{\text{def}}{=} \gcd(\overline{x} - x', N_\lambda)$, and outputs 1 if $p \in \{p_\lambda, q_\lambda\}$, otherwise outputs 0.

By the construction of $\mathcal{D}_3$, we have

$$P_2 = \Pr_{(N_\lambda, r_1', x') \leftarrow \mathcal{S}_1(1^\lambda, N_\lambda)} [\mathcal{D}_3^{(p_\lambda, q_\lambda, N_\lambda)}(1^\lambda, N_\lambda, r_1', x') = 1] \ .$$

As the simulator $\mathcal{S}_1$ is statistically close to the real execution of π, it follows that $|P_2 - P_3|$ is negligible, where

$$P_3 \stackrel{\text{def}}{=} \Pr_{r_1 \leftarrow\!\!\$\, U[\{0,1\}^{3\lambda}], x \leftarrow \mathsf{trans}_1^{\pi}(1^\lambda, N_\lambda, (p_\lambda, q_\lambda); r_1, U[\{0,1\}^2])} [\mathcal{D}_3^{(p_\lambda, q_\lambda, N_\lambda)}(1^\lambda, N_\lambda, r_1, x) = 1] \ .$$

Now we evaluate the last probability P_3. The key fact is that, the random tape for $\mathcal{P}_2$ (denoted here by $r_2 \in \{0,1\}^2$) used when choosing the transcript x in defining the probability P_3 is *independent* of the random tape for $\mathcal{P}_2$ (denoted here by $r'_2 \in \{0,1\}^2$) used in the emulation of the protocol π during the algorithm $\mathcal{D}_3$. Let $y = r_1 \bmod N$. By combining Lemmas 1 and 2, the distribution of y is statistically close to $U[(\mathbb{Z}/N_\lambda\mathbb{Z})^\times]$, therefore we have $y \in \mathsf{QR}_{N_\lambda}$ with probability at least $1/4 - \mathsf{negl}(\lambda)$. On the other hand, assuming $y \in \mathsf{QR}_{N_\lambda}$, the construction of the protocol π implies that both x and $\overline{x}$ (the latter being chosen in the algorithm $\mathcal{D}_3$) are uniformly random square roots of y modulo N_λ, hence are elements of $(\mathbb{Z}/N_\lambda\mathbb{Z})^\times$, and x and $\overline{x}$ are independent due to the independence of r_2 and r'_2 mentioned above. Therefore, we have $x \notin \{\overline{x}, -\overline{x}\}$ with (conditional) probability $1/2$, and once $x \notin \{\overline{x}, -\overline{x}\}$ is satisfied, it follows that $p = \gcd(\overline{x} - x, N_\lambda)$ is a non-trivial divisor of N_λ due to the property $x^2 = y = \overline{x}^2$ modulo N_λ. Hence $\mathcal{D}_3$ outputs 1 in this case. Summarizing, we have $P_3 \geq 1/8 - \mathsf{negl}(\lambda)/2 = \Omega(1)$.

Now recall that all of $|P_\mathcal{F} - P_1|$, $|P_1 - P_2|$, and $|P_2 - P_3|$ are negligible as shown above. Therefore, we have $P_\mathcal{F} = \Omega(1)$ as well as P_3, which means that the algorithm $\mathcal{F}$ outputs a prime factor of $\mathcal{B}(1^\lambda)$ with probability $\Omega(1)$. This completes the proof of Proposition 3. □

Proof of Theorem 1. Owing to Propositions 1 and 2, it suffices to show that the protocol $\pi \circ_1 \mathcal{R}$ is not secure against semi-honest $\mathcal{P}_1$. Now if $\pi \circ_1 \mathcal{R}$ were secure, then Proposition 3 would imply the existence of a PPT uniform algorithm that outputs a prime factor of $\mathcal{B}(1^\lambda)$ with probability $\Omega(1)$, which contradicts the hypothesis that the Blum integer generator $\mathcal{B}$ is not efficiently factorizable. This completes the proof of Theorem 1. □

5 Security When Using a PRG: Second Negative Results

One may feel that our example of a two-party protocol constructed in Section 4, where the use of the given secure PRG leads to an insecure protocol, looks too artificial (for example, the protocol even has no outputs). In this section, we show that a similar phenomenon may still occur for a more realistic protocol. Namely, we show (under a certain computational assumption) that an existing two-party protocol in the literature (proposed by Asharov et al. in ACM CCS 2013 [1]; see Section 5.1) also falls into insecure when one of the two parties uses a certain secure PRG constructed in Section 5.2.

5.1 The Protocol

We focus on a two-party oblivious transfer protocol proposed by Asharov et al. in ACM CCS 2013 [1]; more precisely, we refer to Protocol 51 in Section 5.2 of the full version for the paper [1]. Here we slightly modify the detailed description of the protocol without changing its essential behavior; for example, we explicitly state that the internal randomness for the two parties are bit strings and then utilize approximately uniform sampling of several objects using random bit strings based on Lemmas 1 and 2.

Before going into details of the aforementioned protocol, we note that the input objects for the protocol are classified into global parameters that can be reused for several protocol executions (such as the underlying cyclic group) and "actual" inputs for each individual protocol execution. In fact, the protocol was designed in the original paper [1] for allowing multiple executions of the oblivious transfer using the same global parameter, though this paper deals with the case of a single execution only. For those global parameters, in this paper we put an assumption that a secure global parameter (associated to each security parameter) can be chosen efficiently and *deterministically* (see below for a more precise statement). This technical assumption would also have some practical meaning, since it may sometimes happen that an implementation of a protocol hard-wires such a reusable global parameter.

In order to specify our choice of global parameters, we quote the following description from the text in the second paragraph of Section 5.2 in the full version of [1] (where "[.....]" indicates omission by the author of the present paper):

> [.....] *We also assume that it is possible to sample a random element of the group, and the DDH assumption will remain hard even when the coins used to sample the element are given to the*

distinguisher (i.e., (g, h, g^a, h^a) is indistinguishable from (g, h, g^a, g^b) for random a, b, even given the coins used to sample h). [.....] For finite fields, one can sample a random element $h \in \mathbb{Z}_p$ of order q by choosing a random $x \in_R \mathbb{Z}_p$ and computing $h = x^{(p-1)/q}$ until $h \neq 1$. [.....]

Accordingly, we use the subgroup of a given order q in the multiplicative group $(\mathbb{F}_p)^\times$ of a finite field $\mathbb{F}_p$ (denoted by $\mathbb{Z}_p$ in the quoted text) as the underlying group of the protocol[4], where p is a t-bit prime for some polynomially bounded $t \geq \lambda$ and q is a divisor of $p-1$. Then the sampling method for the group elements indicated in the quoted text above can be realized as the following algorithm $\mathcal{H}$, where slight modification is made in order to ensure that it always halts within finite (polynomial) time.

- Given an input $x' \in \{0,1\}^{2t}$, the algorithm $\mathcal{H}$ computes $x \leftarrow x' \bmod p \in \mathbb{F}_p$, and if $x^{(p-1)/q} \bmod p \notin \{0,1\}$ then it outputs the $x^{(p-1)/q} \bmod p$, or else it outputs 1.

Lemma 6. *The output $\mathcal{H}(x')$ for $x' \leftarrowtail U[\{0,1\}^{2t}]$ is an element of the unique subgroup of order q in $(\mathbb{F}_p)^\times$ and is statistically close to uniform over this subgroup.*

Proof. As $(\mathbb{F}_p)^\times$ is a cyclic group of order $p-1$, $x^{(p-1)/q} \bmod p$ is either 0 or an element of $(\mathbb{F}_p)^\times$ of order dividing q. This proves the former assertion. For the latter assertion, if the x were a uniformly random element of $(\mathbb{F}_p)^\times$ then $x^{(p-1)/q} \bmod p$ would be a uniformly random element of this subgroup. Now Lemmas 1 and 2 imply that the x is statistically close to a uniformly random element of $(\mathbb{F}_p)^\times$ (since $t \geq \lambda$), therefore the latter assertion indeed holds. This completes the proof. □

Now we introduce the aforementioned assumption on the secure and deterministic choice for the global parameters, which is a variant of the decisional Diffie–Hellman (DDH) assumption; cf., Appendix A of the full version for [1][5].

Assumption 1. There exists a deterministic polynomial-time algorithm to choose (given a security parameter 1^λ) a t-bit prime p with $t \geq \lambda$, a divisor q of $p-1$, a generator g of the subgroup of order q in $(\mathbb{F}_p)^\times$, and a deterministic polynomial-time (in λ) key derivation function $\mathsf{KDF}\colon \langle g \rangle \to \{0,1\}^L$ for some L, satisfying the following: The two distributions

$$\left[(p, q, g, g^r \bmod p, x', \mathsf{KDF}(\mathcal{H}(x')^r \bmod p))\colon r \leftarrowtail U[\{0, \ldots, q-1\}], x' \leftarrowtail U[\{0,1\}^{2t}]\right]$$

and

$$\left[(p, q, g, g^r \bmod p, x', z)\colon r \leftarrowtail U[\{0, \ldots, q-1\}], x' \leftarrowtail U[\{0,1\}^{2t}], z \leftarrowtail U[\{0,1\}^{L}]\right]$$

are computationally indistinguishable against non-uniform distinguishers.

Now we give the description of the protocol in Figure 2, where the global parameters associated to security parameter 1^λ are supposed to be chosen deterministically as in Assumption 1. We note that the protocol correctly computes the desired output of Receiver (Party $\mathcal{P}_2$), since

$$u^\alpha = (g^r)^\alpha = (g^\alpha)^r = (h^{(\sigma)})^r = k^{(\sigma)} \text{ in } \mathbb{F}_p$$

and hence $v^{(\sigma)} \oplus \mathsf{KDF}(u^\alpha \bmod p) = v^{(\sigma)} \oplus \mathsf{KDF}(k^{(\sigma)}) = x^{(\sigma)}$.

For the security of the protocol, here we focus only on the security against semi-honest Receiver, which is sufficient for our purpose. We give a security proof below rather than just referring to the original paper [1] in order to clarify the concrete construction of the simulator, which will be relevant in our discussion given in Section 6.

Proposition 4. *Under Assumption 1, the protocol in Figure 2 is secure against semi-honest Receiver with computational simulator.*

[4]The authors of [1] in fact also proposed to use elliptic curve groups, which would make the protocol more efficient. Our choice of the subgroup of $(\mathbb{F}_p)^\times$ here is due to the technical simplicity.

[5]We note that a concrete construction of a key derivation function used in the protocol was not discussed even in the original paper [1]. In the present paper, we just assume that such a key derivation function exists and can be efficiently determined.

Global parameters: t-bit prime p, divisor q of $p-1$, $g \in (\mathbb{F}_p)^\times$ of order q, and a key derivation function $\mathsf{KDF}\colon \mathbb{G} \to \{0,1\}^L$		
(Party $\mathcal{P}_1$ (Sender))	Input: $(x^{(0)}, x^{(1)}) \in \{0,1\}^L$ Random tape: $r_1 \in \{0,1\}^{2t}$	Output: (none)
(Party $\mathcal{P}_2$ (Receiver))	Input: $\sigma \in \{0,1\}$ Random tape: $(r_2', r_2'') \in \{0,1\}^{2t} \times \{0,1\}^{2t}$	Output: $x^{(\sigma)}$

1. Receiver computes $h \leftarrow \mathcal{H}(r_2')$ $(h \in \langle g \rangle)$ by using the algorithm $\mathcal{H}$.
2. Receiver computes $\alpha \leftarrow r_2'' \bmod q$, and sets $(h^{(0)}, h^{(1)}) \leftarrow (g^\alpha \bmod p, h)$ if $\sigma = 0$ and $(h^{(0)}, h^{(1)}) \leftarrow (h, g^\alpha \bmod p)$ if $\sigma = 1$. Then Receiver sends $(h^{(0)}, h^{(1)})$ to Sender.
3. Sender computes $r \leftarrow r_1 \bmod q$, $u \leftarrow g^r \bmod p$, $(k^{(0)}, k^{(1)}) \leftarrow ((h^{(0)})^r \bmod p, (h^{(1)})^r \bmod p)$ and $(v^{(0)}, v^{(1)}) \leftarrow \big(x^{(0)} \oplus \mathsf{KDF}(k^{(0)}), x^{(1)} \oplus \mathsf{KDF}(k^{(1)})\big)$. Then Sender sends u, $v^{(0)}$, and $v^{(1)}$ to Receiver.
4. Receiver outputs $v^{(\sigma)} \oplus \mathsf{KDF}(u^\alpha \bmod p)$; while Sender outputs nothing.

Figure 2: An oblivious transfer protocol from [1]; here we suppose that the global parameters associated to security parameter 1^λ are chosen deterministically as in Assumption 1, and $\oplus$ denotes bit-wise XOR

Proof. First of all, by Assumption 1, the global parameters associated to security parameter 1^λ are efficiently and uniquely determined, therefore those can be ignored when considering the (real or simulated) views for the two parties.

We define a simulator $\mathcal{S}_2 = \mathcal{S}_2(1^\lambda, \sigma, x^{(\sigma)})$ for Receiver as follows:

1. Given 1^λ, σ and $x^{(\sigma)}$ as input, $\mathcal{S}_2$ first chooses a uniformly random tape (r_2', r_2'') for Receiver.
2. $\mathcal{S}_2$ computes $\alpha \leftarrow r_2'' \bmod q$ and $h^{(\sigma)} \leftarrow g^\alpha \bmod p$.
3. $\mathcal{S}_2$ chooses a uniformly random tape r_1 for Sender, and computes $r \leftarrow r_1 \bmod q$, $u \leftarrow g^r \bmod p$, $k^{(\sigma)} \leftarrow (h^{(\sigma)})^r \bmod p$, and $\overline{v}^{(\sigma)} \leftarrow x^{(\sigma)} \oplus \mathsf{KDF}(k^{(\sigma)})$.
4. $\mathcal{S}_2$ chooses $\overline{v}^{(1-\sigma)} \leftarrow\!\!\!\!\circ\, U[\{0,1\}^L]$, and outputs (r_2', r_2'') as the simulated random tape for Receiver and $(u, \overline{v}^{(0)}, \overline{v}^{(1)})$ as the simulated transcript for Receiver.

We assume, for the contrary, that the output of $\mathcal{S}_2$ and Receiver's view in a real protocol execution can be distinguished by a PPT non-uniform distinguisher $\mathcal{D}$ with advice a_λ. Then there are sequences of σ_λ, $x_\lambda^{(0)}$, and $x_\lambda^{(1)}$ satisfying that, the probability

$$P_{\mathrm{real}} \stackrel{\mathrm{def}}{=} \Pr[\mathcal{D}^{(a_\lambda)}(1^\lambda, \sigma_\lambda, r_2', r_2'', u, v^{(0)}, v^{(1)}, x_\lambda^{(\sigma_\lambda)}) = 1]$$

for the objects distributed as in a real protocol execution with Sender's input $(x_\lambda^{(0)}, x_\lambda^{(1)})$ and Receiver's input σ_λ has non-negligible difference from the probability

$$P_{\mathrm{sim}} \stackrel{\mathrm{def}}{=} \Pr[\mathcal{D}^{(a_\lambda)}(1^\lambda, \sigma_\lambda, r_2', r_2'', u, \overline{v}^{(0)}, \overline{v}^{(1)}, x_\lambda^{(\sigma_\lambda)}) = 1]$$

for the objects simulated by $\mathcal{S}_2(1^\lambda, \sigma_\lambda, x_\lambda^{(\sigma_\lambda)})$.

Now we define a non-uniform distinguisher $\overline{\mathcal{D}}$, with advice $\overline{a}_\lambda \stackrel{\mathrm{def}}{=} (a_\lambda, \sigma_\lambda, x_\lambda^{(0)}, x_\lambda^{(1)})$, for two triples $T_0 \stackrel{\mathrm{def}}{=} (g^r \bmod p, x', \mathsf{KDF}(\mathcal{H}(x')^r \bmod p))$ and $T_1 \stackrel{\mathrm{def}}{=} (g^r \bmod p, x', z)$ as in Assumption 1 in the following manner:

1. Given 1^λ and a triple $(u, x', \overline{z})$, $\overline{\mathcal{D}}$ chooses $r_2'' \leftarrow\!\!\!\!\circ\, U[\{0,1\}^{2t}]$ and computes $\alpha \leftarrow r_2'' \bmod q$ and $h^{(\sigma_\lambda)} \leftarrow g^\alpha \bmod p$.

2. $\overline{\mathcal{D}}$ computes $k^{(\sigma_\lambda)} \leftarrow u^\alpha \bmod p$ and $\widetilde{v}^{(\sigma_\lambda)} \leftarrow x_\lambda^{(\sigma_\lambda)} \oplus \mathsf{KDF}(k^{(\sigma_\lambda)})$.

3. $\overline{\mathcal{D}}$ computes $\widetilde{v}^{(1-\sigma_\lambda)} \leftarrow x_\lambda^{(1-\sigma_\lambda)} \oplus \overline{z}$, executes $\mathcal{D}^{(a_\lambda)}(1^\lambda, \sigma_\lambda, x', r_2'', u, \widetilde{v}^{(0)}, \widetilde{v}^{(1)}, x_\lambda^{(\sigma_\lambda)})$, and outputs the output value of the $\mathcal{D}$.

We note that the distinguisher $\overline{\mathcal{D}}$ is PPT as well as $\mathcal{D}$.

By the constructions of $\overline{\mathcal{D}}$ and the protocol, when the input is T_0, the distribution of $(x', r_2'', u, \widetilde{v}^{(0)}, \widetilde{v}^{(1)})$ generated in $\overline{\mathcal{D}}$ is identical to $(r_2', r_2'', u, v^{(0)}, v^{(1)})$ in a real protocol execution except that the element r is given by $r \leftarrow\!\!\!\!\circ\, U[\{0, \ldots, q-1\}]$ in the former, while $r \leftarrow r_1 \bmod q$ with $r_1 \leftarrow\!\!\!\!\circ\, U[\{0,1\}^{2t}]$ in the latter. As $t \geq \lambda$ and the bit length of q is at most t, Lemma 2 implies that the two distributions of r are statistically close. Hence the distributions of $(x', r_2'', u, \widetilde{v}^{(0)}, \widetilde{v}^{(1)})$ and of $(r_2', r_2'', u, v^{(0)}, v^{(1)})$ above are also statistically close, therefore we have

$$|\Pr[\overline{\mathcal{D}}^{(\overline{a}_\lambda)}(1^\lambda, T_0) = 1] - P_{\mathrm{real}}| = \mathsf{negl}(\lambda) \ .$$

On the other hand, when the input is T_1, $z \in \{0,1\}^L$ is uniformly random and is independent of the other objects, therefore $\widetilde{v}^{(1-\sigma_\lambda)}$ is also uniformly random over $\{0,1\}^L$ and is independent of the other objects. This implies that the distribution of $(x', r_2'', u, \widetilde{v}^{(0)}, \widetilde{v}^{(1)})$ generated in $\overline{\mathcal{D}}$ is identical to $(r_2', r_2'', u, \overline{v}^{(0)}, \overline{v}^{(1)})$ generated in $\mathcal{S}_2(1^\lambda, \sigma_\lambda, x_\lambda^{(\sigma_\lambda)})$ except that the element r is given by $r \leftarrow\!\!\!\!\circ\, U[\{0, \ldots, q-1\}]$ in the former, while $r \leftarrow r_1 \bmod q$ with $r_1 \leftarrow\!\!\!\!\circ\, U[\{0,1\}^{2t}]$ in the latter. Now the same argument as above implies that the two distributions of r are statistically close and hence

$$|\Pr[\overline{\mathcal{D}}^{(\overline{a}_\lambda)}(1^\lambda, T_1) = 1] - P_{\mathrm{sim}}| = \mathsf{negl}(\lambda) \ .$$

Since $|P_{\mathrm{real}} - P_{\mathrm{sim}}|$ is non-negligible as mentioned above, it follows that $\Pr[\overline{\mathcal{D}}^{(\overline{a}_\lambda)}(1^\lambda, T_0) = 1]$ and $\Pr[\overline{\mathcal{D}}^{(\overline{a}_\lambda)}(1^\lambda, T_1) = 1]$ also have non-negligible difference. This contradicts the hypothesis in Assumption 1 that T_0 and T_1 are computationally indistinguishable against non-uniform distinguishers. As a result, it follows that the output of $\mathcal{S}_2$ and Receiver's view in a real protocol execution are computationally indistinguishable. This completes the proof of Proposition 4. □

5.2 The PRG

Based on the protocol described in Section 5.1, we can establish the following result which is analogous to Theorem 1. For the result, we also put another technical assumption as follows:

Assumption 2. In the situation of Assumption 1, the parameters can be chosen in a way that $(p-1)/q$ is coprime to q, and a generator g_0 of $(\mathbb{F}_p)^\times$ can also be chosen in deterministic polynomial time (in λ).

Then our result is stated as follows:

Theorem 2. *Suppose that Assumptions 1 and 2 are true. Assume moreover that a secure PRG exists. Then there exist a two-party protocol π and a* secure *PRG $\mathcal{R}$ with the following two properties:*

- *π is secure against semi-honest Party $\mathcal{P}_2$.*
- *$\pi \circ_2 \mathcal{R}$ is* not *secure against semi-honest Party $\mathcal{P}_2$.*

The protocol π indicated in the statement of the theorem is the one described in Section 5.1, which is secure against semi-honest Party $\mathcal{P}_2$ (Receiver) as in Proposition 4. From now, we construct the PRG $\mathcal{R}$ indicated in the statement of the theorem.

Recall that the central idea of the protocol π was to let Receiver sample, by using the algorithm $\mathcal{H}$, a random element h of the cyclic group $\langle g \rangle$ in a way that Receiver cannot know the discrete logarithm of h with respect to g even if Receiver can see the internal randomness used to sample the h. Intuitively, our construction of the PRG $\mathcal{R}$ is intended to disable the functionality of $\mathcal{H}$ for concealing the discrete logarithm.

To construct the PRG, first we define an algorithm $\mathcal{R}^\dagger = \mathcal{R}^\dagger(1^\lambda, s)$ with random seed $s = (s_1, s_2, s_3, s_4) \in \{0,1\}^{2t} \times \{0,1\}^{2t} \times \{0,1\}^{3t} \times \{0,1\}^{2t}$. Recall from Assumptions 1 and 2 that the global parameters p, q, g,

and KDF, as well as a generator g_0 of $(\mathbb{F}_p)^\times$, can be deterministically chosen in polynomial time (in λ), and that $(p-1)/q$ is coprime to q. We fix those parameters in the following construction. Now we define $\mathcal{R}^\dagger$ as follows:

1. Given 1^λ and $s = (s_1, s_2, s_3, s_4) \in \{0,1\}^{2t} \times \{0,1\}^{2t} \times \{0,1\}^{3t} \times \{0,1\}^{2t}$, $\mathcal{R}^\dagger$ computes the multiplicative inverse d of $(p-1)/q$ modulo q.
2. $\mathcal{R}^\dagger$ computes $e \leftarrow s_1 \bmod q$ and $h^\dagger \leftarrow g^e \bmod p$.
3. $\mathcal{R}^\dagger$ computes $e' \leftarrow s_2 \bmod (p-1)$ and $h^{\dagger\dagger} \leftarrow (h^\dagger)^d \cdot {g_0}^{qe'} \bmod p$.
4. $\mathcal{R}^\dagger$ computes $r^\dagger \leftarrow h^{\dagger\dagger} + (s_3 \bmod K) \cdot p \in \{0, \ldots, 2^{2t}-1\}$ where $K = \lfloor (2^{2t} - 1 - h^{\dagger\dagger})/p \rfloor + 1$, identifies the $r^\dagger$ with a $2t$-bit sequence, and then outputs the pair $(r^\dagger, s_4) \in \{0,1\}^{2t} \times \{0,1\}^{2t}$.

Proposition 5. *For $s = (s_1, s_2, s_3, s_4) \leftarrowtail U[\{0,1\}^{2t} \times \{0,1\}^{2t} \times \{0,1\}^{3t} \times \{0,1\}^{2t}]$, the output distribution of $\mathcal{R}^\dagger(1^\lambda, s)$ is statistically close to $U[\{0,1\}^{2t} \times \{0,1\}^{2t}]$, and we have $\mathcal{H}(r^\dagger \bmod p) = g^e \bmod p$ where e and $r^\dagger$ are as computed in $\mathcal{R}^\dagger$.*

Proof. For the latter assertion, we have $r^\dagger \bmod p = h^{\dagger\dagger}$ and

$$(h^{\dagger\dagger})^{(p-1)/q} = (h^\dagger)^{d\cdot(p-1)/q} \cdot {g_0}^{qe'\cdot(p-1)/q} = h^\dagger \cdot {g_0}^{e'(p-1)} = h^\dagger = g^e \text{ in } \mathbb{F}_p$$

since $h^\dagger \in \langle g \rangle$ and $d \cdot (p-1)/q \equiv 1 \pmod q$. Now if $g^e \neq 1$ in $\mathbb{F}_p$, then we have $\mathcal{H}(r^\dagger \bmod p) = g^e \bmod p$ by the definition of $\mathcal{H}$. On the other hand, if $g^e = 1$ in $\mathbb{F}_p$, then we have $e = 0$ since g is of order q, while now $\mathcal{H}(r^\dagger \bmod p) = 1$ by the definition of $\mathcal{H}$. Hence we have $\mathcal{H}(r^\dagger \bmod p) = g^e \bmod p$ in any case.

For the former assertion, it suffices to show that $r^\dagger \stackrel{\text{stat}}{\equiv} U[\{0,1\}^{2t}]$. Let $f \in \{0, \ldots, p-2\}$ be the discrete logarithm of g with respect to g_0. Then f is a multiple of $(p-1)/q$ since $g^q = 1$ in $\mathbb{F}_p$; we put $f = f'(p-1)/q$ with $1 \leq f' \leq q-1$. Now both f' and $(p-1)/q$ are coprime to q, so is f.

Since $s_1, s_2 \in \{0,1\}^{2t}$ are sufficiently long, by virtue of Lemma 2, we may assume without loss of generality that $e \leftarrowtail U[\{0, \ldots, q-1\}]$ and $e' \leftarrowtail U[\{0, \ldots, p-2\}]$. Now we have $h^{\dagger\dagger} = g^{ed} \cdot {g_0}^{qe'} = {g_0}^{fed+qe'}$ in $\mathbb{F}_p$. Since $fed + qe' \bmod q = e \cdot fd \bmod q$ and fd is coprime to q by the argument above, it follows that $fed + qe' \bmod q$ is uniformly random as well as e. On the other hand, since $\lfloor (fed + qe')/q \rfloor = e' + \lfloor fed/q \rfloor$, it follows from the uniformly random choice of e' that $\lfloor (fed + qe')/q \rfloor \bmod (p-1)/q$ is uniformly random and is independent of $fed + qe' \bmod q = fed \bmod q$. These arguments imply that $fed + qe' \bmod (p-1)$ is uniformly random, therefore $h^{\dagger\dagger} = {g_0}^{fed+qe'}$ is also uniformly random over $(\mathbb{F}_p)^\times$.

Since $s_3 \in \{0,1\}^{3t}$ is sufficiently long, Lemma 3 implies that the conditional distribution of $r^\dagger$ conditioned on a given $h^{\dagger\dagger}$ is statistically close to the uniform distribution over the set of all $r'_2 \in \{0,1\}^{2t}$ with $r'_2 \bmod p = h^{\dagger\dagger}$. Now if the distribution of $h^{\dagger\dagger}$ were identical to the distribution of $r'_2 \bmod p$ with $r'_2 \leftarrowtail U[\{0,1\}^{2t}]$, then the distribution of $r^\dagger$ would be statistically close to $U[\{0,1\}^{2t}]$ by the argument above. Moreover, since p has bit length $t \geq \lambda$ and $r'_2 \in \{0,1\}^{2t}$, Lemma 3 implies that the distributions of both $h^{\dagger\dagger}$ and $r'_2 \bmod p$ are statistically close to $U[\mathbb{F}_p]$, hence the two distributions themselves are statistically close to each other. By these arguments, it follows that the distribution of $r^\dagger$ is indeed statistically close to $U[\{0,1\}^{2t}]$. This completes the proof of Proposition 5. □

On the other hand, there exists a secure PRG by the hypothesis of Theorem 2. As $2t + 2t + 3t + 2t = 9t$ is polynomially bounded in λ, a standard technique to securely expand the output length of a PRG (see e.g., Section 3.3 of [6]) yields a secure PRG $\mathcal{R}_0$ with stretch function $\ell_0(\lambda) = 9t$. Now we define the PRG $\mathcal{R}$ as the composition of $\mathcal{R}^\dagger$ and $\mathcal{R}_0$; $\mathcal{R}(1^\lambda, \overline{s}) = \mathcal{R}^\dagger(1^\lambda, \mathcal{R}_0(1^\lambda, \overline{s}))$ for $\overline{s} \in \{0,1\}^\lambda$. Then by essentially the same argument as Proposition 2, the security of $\mathcal{R}_0$ combined with Proposition 5 implies the following result.

Proposition 6. *The PRG $\mathcal{R}$ is secure.*

Proof of Theorem 2. It suffices now to show that the protocol $\pi \circ_2 \mathcal{R}$ is not secure against semi-honest Party $\mathcal{P}_2$ (Receiver). We note that, if the protocol were secure, then given a local input σ and a local output $x^{(\sigma)}$, Receiver should not be able to distinguish Receiver's views during protocol executions for different choices

of Server's secret input $x^{(1-\sigma)}$. However, when using the output $(r^\dagger, s_4)$ of $\mathcal{R}(1^\lambda, \overline{s})$ with known seed $\overline{s}$ as Receiver's random tape in π, Receiver can compute from the seed $\overline{s}$ as in Proposition 5 the discrete logarithm e of the element $h = \mathcal{H}(r^\dagger \bmod p) = g^e \bmod p$ used in the protocol. This enables Receiver to extract the other secret input $x^{(1-\sigma)}$ of Sender from the received messages u and $v^{(1-\sigma)}$ as $x^{(1-\sigma)} = v^{(1-\sigma)} \oplus \mathsf{KDF}(u^e \bmod p)$, since $k^{(1-\sigma)} = (h^{(1-\sigma)})^r = h^r = (g^e)^r = (g^r)^e = u^e$ in $\mathbb{F}_p$. This means that the protocol $\pi \circ_2 \mathcal{R}$ is not secure against semi-honest Receiver, completing the proof of Theorem 2. □

6 Security When Using a PRG: Positive Result

Due to the results in Sections 4 and 5, an intuitive expectation, that a secure two-party protocol combined with a secure PRG would also be secure, is not unconditionally true. In this section, we give a sufficient condition for a two-party protocol and a PRG to ensure that their combination is still secure.

A part of the proposed condition for a two-party protocol concerns a certain structure of a simulator constructed in a security proof of the protocol. More precisely, we introduce the following definition.

Definition 4. We say that a simulator $\mathcal{S}_i$ for Party $\mathcal{P}_i$ in a two-party protocol is *with raw random tape*, if $\mathcal{S}_i$ is executed in the following manner with some PPT algorithm $\mathcal{T}_i$:

- Given 1^λ, x_i and $f_i(\vec{x})$ as inputs, $\mathcal{S}_i$ first generates a uniformly random tape r_i for Party $\mathcal{P}_i$, and then executes $\mathcal{T}_i(1^\lambda, x_i, f_i(\vec{x}), r_i)$ to obtain the remaining part of a simulated view of Party $\mathcal{P}_i$.

Let $\langle r_i, V_i \rangle$, where $V_i = \mathcal{T}_i(1^\lambda, x_i, f_i(\vec{x}), r_i)$, denote the simulated view for Party $\mathcal{P}_i$ consisting of the random tape r_i and the remaining part V_i (we suppose that the components in $\langle r_i, V_i \rangle$ are appropriately reordered to keep consistency with the syntax in Definition 3).

Intuitively, the definition means that, for the random tape part of the simulated view, the simulator just outputs an ideally sampled random tape *as is* (which is then used for simulating the transcript), rather than using an artificially adjusted random tape generated from a simulated transcript. For example, the simulator $\mathcal{S}_2$ constructed in the proof of Proposition 4 is in fact with raw random tape in this sense, while the simulator $\mathcal{S}_1$ in the proof of Proposition 1 is not.

From now, we give a result as mentioned above. Before stating the result, we recall that the *min-entropy* of a random variable X is defined by $H_\infty(X) = -\max_x \log_2 \Pr[X = x]$. Then we have the following result:

Theorem 3. *Let π be a two-party protocol, $i \in \{1, 2\}$, and let $\mathcal{R}$ be a PRG with stretch function ℓ to generate the random tape for Party $\mathcal{P}_i$. Suppose that the following three conditions are satisfied:*

1. *π is secure against semi-honest Party $\mathcal{P}_i$ with statistical simulator $\mathcal{S}_i$.*
2. *The simulator $\mathcal{S}_i$ above is with raw random tape.*
3. *For uniformly random seed for $\mathcal{R}$, we have $\ell(\lambda) - H_\infty(\mathcal{R}(1^\lambda)) = O(\log \lambda)$.*

Then the protocol $\pi \circ_i \mathcal{R}$ is also secure against semi-honest $\mathcal{P}_i$ with statistical simulator with raw random tape.

Proof. By Condition 2 in the statement, there is a PPT algorithm $\mathcal{T}_i$ inside the simulator $\mathcal{S}_i$ as in Definition 4. Let $J_{\mathrm{dom}} = \{0,1\}^\lambda$ and $J_{\mathrm{ran}} = \{0,1\}^{\ell(\lambda)}$ denote the domain and the range of the PRG $\mathcal{R}$, respectively.

Given 1^λ, an input pair $\vec{x} = (x_1, x_2)$, a local output o_i of Party $\mathcal{P}_i$, and a random tape $r_i \in J_{\mathrm{ran}}$ for $\mathcal{P}_i$ in the protocol π, the simulated view for $\mathcal{P}_i$ in π is given by $\langle r_i, \mathcal{T}_i(1^\lambda, x_i, o_i, r_i) \rangle$ (see Definition 4 for the notation $\langle r_i, \cdot \rangle$). On the other hand, let $V_{\mathrm{real}}(1^\lambda, \vec{x}, r_i)$ denote the random variable of the part of the view for $\mathcal{P}_i$ except for the random tape r_i in a real execution of protocol π with input pair $\vec{x}$ and random tape r_i for $\mathcal{P}_i$. Then the view for $\mathcal{P}_i$ in a real execution of π is given by $\langle r_i, V_{\mathrm{real}}(1^\lambda, \vec{x}, r_i) \rangle$.

We define a simulator $\widetilde{\mathcal{S}}_i$ for $\mathcal{P}_i$ in the protocol $\pi \circ_i \mathcal{R}$ as follows: $\widetilde{\mathcal{S}}_i$ chooses $\widetilde{r}_i \leftarrowtail U[J_{\mathrm{dom}}]$, computes $r_i = \mathcal{R}(1^\lambda, \widetilde{r}_i)$, and outputs $\langle \widetilde{r}_i, \mathcal{T}_i(1^\lambda, x_i, o_i, r_i) \rangle$. This simulator is with raw random tape by the construction.

Note that the view for $\mathcal{P}_i$ in a real execution of $\pi \circ_i \mathcal{R}$ is given by $\langle \widetilde{r}_i, V_{\mathrm{real}}(1^\lambda, \vec{x}, r_i)\rangle$. Now let Δ and $\widetilde{\Delta}$ denote the statistical distances between the real and simulated views for $\mathcal{P}_i$ in π and in $\pi \circ_i \mathcal{R}$, respectively, for given 1^λ, $\vec{x} = (x_1, x_2)$, and o_i. Then we have

$$\begin{aligned}
\widetilde{\Delta} &= \frac{1}{2} \sum_{\widetilde{s}_i \in J_{\mathrm{dom}}, V_i} \left| \Pr[\langle \widetilde{r}_i, \mathcal{T}_i(1^\lambda, x_i, o_i, \mathcal{R}(1^\lambda, \widetilde{r}_i))\rangle = \langle \widetilde{s}_i, V_i\rangle] - \Pr[\langle \widetilde{r}_i, V_{\mathrm{real}}(1^\lambda, \vec{x}, \mathcal{R}(1^\lambda, \widetilde{r}_i))\rangle = \langle \widetilde{s}_i, V_i\rangle] \right| \\
&= \frac{1}{2} \sum_{\widetilde{s}_i \in J_{\mathrm{dom}}, V_i} \left| \frac{1}{|J_{\mathrm{dom}}|} \Pr[\mathcal{T}_i(1^\lambda, x_i, o_i, \mathcal{R}(1^\lambda, \widetilde{s}_i)) = V_i] - \frac{1}{|J_{\mathrm{dom}}|} \Pr[V_{\mathrm{real}}(1^\lambda, \vec{x}, \mathcal{R}(1^\lambda, \widetilde{s}_i)) = V_i] \right| \\
&= \frac{1}{2|J_{\mathrm{dom}}|} \sum_{s_i \in J_{\mathrm{ran}}, V_i} \sum_{\substack{\widetilde{s}_i \in J_{\mathrm{dom}} \\ \mathcal{R}(1^\lambda, \widetilde{s}_i) = s_i}} \left| \Pr[\mathcal{T}_i(1^\lambda, x_i, o_i, s_i) = V_i] - \Pr[V_{\mathrm{real}}(1^\lambda, \vec{x}, s_i) = V_i] \right| \ .
\end{aligned}$$

For the second sum in the right-hand side, given $s_i \in J_{\mathrm{ran}}$, the number of $\widetilde{s}_i \in J_{\mathrm{dom}}$ satisfying $\mathcal{R}(1^\lambda, \widetilde{s}_i) = s_i$ is at most $|J_{\mathrm{dom}}| \cdot 2^{-H_\infty(\mathcal{R}(1^\lambda))}$ where the min-entropy of $\mathcal{R}(1^\lambda)$ is with respect to a uniformly random seed. This implies that

$$\begin{aligned}
\widetilde{\Delta} &\le \frac{1}{2|J_{\mathrm{dom}}|} \cdot |J_{\mathrm{dom}}| \cdot 2^{-H_\infty(\mathcal{R}(1^\lambda))} \sum_{s_i \in J_{\mathrm{ran}}, V_i} \left| \Pr[\mathcal{T}_i(1^\lambda, x_i, o_i, s_i) = V_i] - \Pr[V_{\mathrm{real}}(1^\lambda, \vec{x}, s_i) = V_i] \right| \\
&= \frac{1}{2} \cdot 2^{-H_\infty(\mathcal{R}(1^\lambda))} \sum_{s_i \in J_{\mathrm{ran}}, V_i} \left| \Pr[\mathcal{T}_i(1^\lambda, x_i, o_i, s_i) = V_i] - \Pr[V_{\mathrm{real}}(1^\lambda, \vec{x}, s_i) = V_i] \right| \ .
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\Delta &= \frac{1}{2} \sum_{s_i \in J_{\mathrm{ran}}, V_i} \left| \Pr[\langle r_i, \mathcal{T}_i(1^\lambda, x_i, o_i, r_i)\rangle = \langle s_i, V_i\rangle] - \Pr[\langle r_i, V_{\mathrm{real}}(1^\lambda, \vec{x}, r_i)\rangle = \langle s_i, V_i\rangle] \right| \\
&= \frac{1}{2} \sum_{s_i \in J_{\mathrm{ran}}, V_i} \left| \frac{1}{|J_{\mathrm{ran}}|} \Pr[\mathcal{T}_i(1^\lambda, x_i, o_i, s_i) = V_i] - \frac{1}{|J_{\mathrm{ran}}|} \Pr[V_{\mathrm{real}}(1^\lambda, \vec{x}, s_i) = V_i] \right| \\
&= \frac{1}{2|J_{\mathrm{ran}}|} \sum_{s_i \in J_{\mathrm{ran}}, V_i} \left| \Pr[\mathcal{T}_i(1^\lambda, x_i, o_i, s_i) = V_i] - \Pr[V_{\mathrm{real}}(1^\lambda, \vec{x}, s_i) = V_i] \right| \ .
\end{aligned}$$

Hence we have

$$\widetilde{\Delta} \le \frac{1}{2} \cdot 2^{-H_\infty(\mathcal{R}(1^\lambda))} \cdot 2|J_{\mathrm{ran}}| \cdot \Delta = 2^{\log_2 |J_{\mathrm{ran}}| - H_\infty(\mathcal{R}(1^\lambda))} \cdot \Delta \ .$$

By Condition 1 in the statement, Δ is bounded by a common negligible function $\mathsf{negl}(\lambda)$. On the other hand, we have $|J_{\mathrm{ran}}| = 2^{\ell(\lambda)}$ and $\log_2 |J_{\mathrm{ran}}| = \ell(\lambda)$, and by Condition 3 in the statement, $2^{\ell(\lambda) - H_\infty(\mathcal{R}(1^\lambda))} = 2^{O(\log \lambda)}$ is polynomially bounded. This implies that $\widetilde{\Delta}$ is also negligible, therefore the output of $\widetilde{\mathcal{S}}_i$ is statistically close to the view in the real protocol execution. This completes the proof of Theorem 3. □

We give some remarks on the conditions in Theorem 3. First, for Condition 3, if the PRG $\mathcal{R}$ is injective as a map from $J_{\mathrm{dom}} = \{0,1\}^\lambda$ to $J_{\mathrm{ran}} = \{0,1\}^{\ell(\lambda)}$, then we have $H_\infty(\mathcal{R}(1^\lambda)) = \lambda$ and hence the condition says that the PRG has logarithmic stretch $\ell(\lambda) - \lambda = O(\log \lambda)$. On the other hand, for the remaining two conditions, we note that the insecure example in Section 5 does not satisfy Condition 1 (while satisfying Condition 2), and the insecure example in Section 4 does not satisfy Condition 2 (while satisfying Condition 1). This suggests that, if we want to generalize Theorem 3 by relaxing Condition 1 or Condition 2, then we would require some alternative condition for structures of the protocol and the PRG[6].

[6]Although these counterexamples in Sections 4 and 5 do not satisfy Condition 3 about the stretch function of the PRG, this is not an essential point for the failure of the conclusion of Theorem 3. Namely, the PRGs for both of the examples were constructed by first defining a pseudorandom function with statistically close to uniform output distribution, which has a longer input, and then generating the longer input by using another secure PRG with short seed. The seed length was set to λ, which is much shorter than the original random tape for the party, purely due to the consistency with our syntax for PRGs in Definition 2. In the construction of those PRGs, we may adjust the seed length to be just 1-bit shorter than the original random tape, and the insecurity result on the protocol obtained by applying the PRG still holds even for such a PRG with 1-bit stretch.

Acknowledgments. The author thanks all the members of a study group "Shin-Akarui-Angou-Benkyoukai" for fruitful discussions on the results of this paper. Among them, the author is most grateful to Shota Yamada and Tadanori Teruya for his smart advice on this study, and the author also specially thanks Kazumasa Shinagawa, Takashi Yamakawa, Takahiro Matsuda, Yusuke Sakai, Keita Emura, and Goichiro Hanaoka for their precious comments. Most of this work was supported by JST PRESTO Grant Number JPMJPR14E8, Japan.

References

[1] G. Asharov, Y. Lindell, T. Schneider, M. Zohner: More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In: Proceedings of ACM CCS 2013, pp.535–548, 2013. Full version available at IACR Cryptology ePrint Archive 2013/552 (`https://eprint.iacr.org/2013/552`), 2013 (Version 20130904:141912).

[2] J. P. Degabriele, K. G. Paterson, J. C. N. Schuldt, J. Woodage: Backdoors in Pseudorandom Number Generators: Possibility and Impossibility Results. In: Proceedings of CRYPTO 2016 (Part I), LNCS vol.9814, pp.403–432, 2016.

[3] Y. Dodis, C. Ganesh, A. Golovnev, A. Juels, T. Ristenpart: A Formal Treatment of Backdoored Pseudorandom Generators. In: Proceedings of EUROCRYPT 2015 (Part I), LNCS vol.9056, pp.101–126, 2015.

[4] Y. Dodis, S. J. Ong, M. Prabhakaran, A. Sahai: On the (Im)possibility of Cryptography with Imperfect Randomness. In: Proceedings of FOCS 2004, pp.196–205, 2004.

[5] Y. Dodis, Y. Yao: Privacy with Imperfect Randomness. In: Proceedings of CRYPTO 2015 (Part II), LNCS vol.9216, pp.463–482, 2015.

[6] O. Goldreich: Foundations of Cryptography, Volume I. Cambridge University Press, 2001.

[7] O. Goldreich: Foundations of Cryptography, Volume II. Cambridge University Press, 2004.

[8] C. Hazai, H. Zarosim: The Feasibility of Outsourced Database Search in the Plain Model. In: Proceedings of SCN 2016, LNCS vol.9841, pp.313–332, 2016.

[9] N. Heninger, Z. Durumeric, E. Wustrow, J. A. Halderman: Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In: Proceedings of USENIX Security Symposium 2012, pp.205–220, 2012.

[10] P. Hubácek, D. Wichs: On the Communication Complexity of Secure Function Evaluation with Long Output. In: Proceedings of ITCS 2015, pp.163–172, 2015. Full version available at IACR Cryptology ePrint Archive 2014/669 (`http://eprint.iacr.org/2014/669`), 2014 (Version 20140828:224736).

[11] Y. Lindell: How To Simulate It – A Tutorial on the Simulation Proof Technique. IACR Cryptology ePrint Archive 2016/046 (`http://eprint.iacr.org/2016/046`), 2016 (Version 20160524:061302).

[12] Y Lindell, K. Nissim, C. Orlandi: Hiding the Input-Size in Secure Two-Party Computation. In: Proceedings of ASIACRYPT 2013 (Part II), LNCS vol.8270, 421–440, 2013.

[13] Y Lindell, K. Nissim, C. Orlandi: Hiding the Input-Size in Secure Two-Party Computation. IACR Cryptology ePrint Archive 2012/679 (`http://eprint.iacr.org/2012/679`), 2012 (Version 20160401:113657).

[14] M. O. Rabin: Digitalized Signatures and Public-Key Functions as Intractable as Factorization. MIT Laboratory for Computer Science Technical Report, 1979.

[15] K. Shinagawa, K. Nuida, T. Nishide, G. Hanaoka, E. Okamoto: Size-Hiding Computation for Multiple Parties. In: ASIACRYPT 2016 (Part II), LNCS 10032, pp.937–966, 2016.