Semi-Honest Secure Multiparty Computation Can Be Insecure with Use of Even Almost Uniformly Random Number Generators

Koji Nuida[1,2]

[1] National Institute of Advanced Industrial Science and Technology (AIST), Japan
(k.nuida@aist.go.jp)
[2] Japan Science and Technology Agency (JST) PRESTO

July 28, 2017

Abstract

It is widely understood that we are just human beings rather than being almighty; as a result, we have always to rely on *non-ideal* randomness in real-world cryptosystems. In contrast, in a theoretical design of a cryptosystem, it is usually regarded as sufficient to give a security proof assuming the use of *ideal* randomness in the system. This gap between the theory and the real world seems to have not been considered seriously, due to our common beliefs that computationally secure (close-to-ideal) randomness generation must be practically possible and the security of cryptosystems must be preserved when the ideal randomness in theory is replaced by a practical computationally secure pseudorandom generator (PRG). Nevertheless, in this paper we reveal, as opposed to our intuition, that the latter kind of common belief on the security under the use of computationally secure PRGs is NOT true for a general two-party computation protocol. More precisely, we prove as a counterexample that an oblivious transfer protocol proposed by Asharov et al. in ACM CCS 2013 in the semi-honest model becomes *insecure* when a party in the protocol uses, to generate his/her internal randomness, an even *statistically* secure PRG constructed in the present paper. Our result implies that, even by using a statistically secure PRG implemented with a best effort by an honest engineer, the security of an implemented two-party protocol is not automatically guaranteed; the author thinks that this fact should be widely recognized in the area of cryptography as well as many other areas where secure multiparty computation is going to be practically applied. It might be an interesting future research topic to study sufficient conditions for two-party protocols that imply the security under a use of a secure PRG, a part of which is also done in this paper.

1 Introduction

In the theory and applications of cryptography, the gap between the ideal random numbers in theoretical models and the non-ideal random numbers in the real world is always a major worrying problem. For example, Heninger et al. revealed in 2012 [10] 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. This is in fact a sort of "worst" examples; but in general, it seems to be even beyond the art of human being to generate a perfectly ideal random number. Accordingly, for establishing secure cryptographic systems in the real world, we have always been using random numbers that are non-ideal to various extent; e.g., computationally indistinguishable from ideal, (seemingly) statistically close to ideal, or sometimes even more casual "non-cryptographic" pseudorandom generators (PRGs) such as the Mersenne Twister [15].

When a theoretical cryptographer proposes a new cryptographic scheme in an academic paper, he/she usually gives a mathematical proof showing that the scheme is secure (in a certain, formally defined sense) provided the scheme is implemented and used precisely as specified by the description in the paper (and, often moreover, if some cryptographic assumption holds such as the computational hardness of the integer factoring). On the other hand, he/she usually guarantees nothing about the security in cases where some part

of the specification for the construction of his/her scheme is not precisely followed by a real implementation of the scheme. Accordingly, in order to benefit from the security proof in the paper, an engineer who wants to implement the proposed scheme has to precisely follow the specification in the paper; and it seems to be an ordinary custom in the area of cryptography that, when an implemented real system does not follow the specification precisely and then fails the security claimed in the paper, this is regarded as the responsibility of the engineer rather than the cryptographer who proposed the scheme.

From the viewpoint of human being's limitation for randomness generation discussed in the first paragraph, the aforementioned custom in the cryptographic area might look unfavorable for the engineer's side. Namely, most of the cryptographic papers to propose a new scheme are assuming that ideal random numbers are used in the scheme, while it is practically infeasible for the engineers to precisely follow the specification for the ideal random numbers. This kind of unfairness in the custom has, however, not been considered seriously so far due to the following two common beliefs in the area of cryptography:

1. It must be practically feasible (possibly under some cryptographic assumption) to generate randomness that is not ideal but still *computationally indistinguishable* from the ideal randomness.

2. It must be always true that, given a cryptographic scheme that is computationally secure when using ideal randomness, the scheme remains secure after replacing the ideal randomness by an output of a PRG that is computationally indistinguishable from the ideal randomness.

If these are certainly true, it is guaranteed to be sufficient for the engineers to use such a computationally secure PRG instead of generating the ideal randomness in order to securely implement a cryptographic scheme, while the author of a paper to propose a new cryptographic scheme is allowed to only consider the use of the ideal randomness in the same way as the present time. Actually, the standard formalization of the computational indistinguishability for an output of a PRG from the ideal randomness seems to be intended to make sure the latter belief above; and the paradigm to bridge a gap between the theory and the implementation of cryptographic schemes based on the two beliefs above seems to be working well for most of the situations in cryptography.

Nevertheless, here we recall that there are no formal proofs to guarantee that the second common belief above is indeed true for an arbitrary cryptographic scheme. And in fact, the aim of the present paper is to reveal that, as opposed to the usual cryptographers' intuition, *the second common belief above is NOT always true for some kind of cryptographic schemes*, particularly for the case of *secure two-party computation*. Briefly summarizing, a secure 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 any information implied by the function value itself) is kept secret against the other party during the computation.

1.1 Our Results

Briefly summarizing, our main result (Theorem 1 in Section 4) shows that there exist a two-party computation protocol $\pi^\dagger$ that is (computationally) secure in the semi-honest model and a secure PRG $\mathcal{R}^\dagger$ with the following property: *If a party in the protocol $\pi^\dagger$ uses a random output of the PRG $\mathcal{R}^\dagger$ as the party's internal randomness instead of the ideal internal randomness, then the resulting protocol becomes not secure against the party him/herself.* This means that the second common belief in the area of cryptography mentioned above is not true for the case of secure two-party computation in the semi-honest model. It is even worse that the PRG $\mathcal{R}^\dagger$ in our counterexample is not just computationally but *statistically* secure, but such a significantly strong PRG is not yet sufficient for preventing the counter-intuitive security loss. We also emphasize that the two-party protocol $\pi^\dagger$ in our counterexample (see Section 4.1) is chosen from an oblivious transfer protocol proposed by Asharov et al. in ACM CCS 2013 [3], rather than artificially designed only to prove our theorem. See Section 4 for the details of our main result; in the rest of this section, we intuitively explain why such a counterexample exists for the case of two-party computation.

The most noticeable characteristic of two-party computation from the viewpoint of causing such an issue discussed in this paper is that, the entity who uses an internal randomness may be the same as the entity who attacks against the protocol; namely, a two-party protocol is required to be secure against any party

joining a protocol execution rather than a third-party attacker[1]. As a result, the security of two-party protocols has to be investigated under the assumption that an attacker can even see the internal randomness itself used in the protocol (since the random numbers may be stored in the attacker's own device used in a protocol execution). Accordingly, in the case where a (corrupted) party uses a PRG to generate the internal randomness for the protocol from a random seed, the security after the use of the PRG has to be considered by assuming that an attacker can see the random seed for the PRG as well (which may be also stored in the attacker's device). On the other hand, the standard formulation for the security of PRGs is established under the assumption that an attacker cannot directly see the random seed used by the PRG. Therefore, even if the PRG is secure, the security of the PRG (assuming the seed is not visible) cannot in general help for preserving the security for a two-party protocol (in the situation where the seed is visible).

In the two-party protocol $\pi^\dagger$ in our counterexample, the two parties are supposed to agree, before a protocol execution, on some public global parameter including a cyclic group $\mathbb{G} = \langle g \rangle$ of known order q with generator g; and then start the protocol with their local inputs. The central idea for the protocol is that, in some kinds of groups $\mathbb{G}$, one can efficiently sample a random element $h \in \mathbb{G}$ in a way that the discrete logarithm of h with respect to the generator g cannot be determined *even if the randomness used to sample the h is known*. The construction of the protocol $\pi^\dagger$ requires to use such a sampling algorithm for $h \in \mathbb{G}$. Then, by closely analyzing an instantiation (described in the original paper [3]) of the sampling algorithm, we construct a PRG $\mathcal{R}^\dagger$ satisfying that for a uniformly random seed s for $\mathcal{R}^\dagger$, the distribution of $\mathcal{R}^\dagger(s)$ is statistically close to the distribution of the ideal internal randomness for the sampling algorithm, and when $\mathcal{R}^\dagger(s)$ is used as the internal randomness for the sampling algorithm, the discrete logarithm of the sampled $h \in \mathbb{G}$ with respect to g can be efficiently recovered from the seed s. Hence the security of $\pi^\dagger$ is compromised when the PRG $\mathbb{G}$ is used[2]. See Section 4.2 for the concrete construction of the PRG $\mathcal{R}^\dagger$.

Our negative result explained above would lead us to the following question: *How can we guarantee the security of a two-party protocol in the real world where the ideal randomness is not available?* A possible solution is to use a tamper-resistant physical random number generator with close-to-uniform output distribution; now even a semi-honest party him/herself cannot see the internal state for the random number generator and hence the aforementioned negative result may be ignored. Another possibility is to prove the security of the original two-party protocol under a stronger security model that also covers modification of randomness, such as the full malicious model and a somewhat weaker "semi-malicious model" introduced by Asharov, Jain, and Wichs in Section A.2 of [2]. On the other hand, for a more "efficient" solution without a special physical device or a stronger security model (which is significantly more intricate than the semi-honest model), we give an affirmative result (Theorem 2 in Section 5) to prove that the security (in the semi-honest model) of a two-party protocol is preserved by replacing a party's internal randomness with an output of a *statistically* secure PRG, if a simulator can efficiently simulate the party's view in a real execution of the protocol in a *statistically* indistinguishable manner, and the simulator simulates the internal randomness for the party by using some uniformly sampled randomness *as is* (rather than, as frequently done in the security proofs for two-party protocols in the literature, first simulating the other part of the view and then adjusting the simulated internal randomness according to the already chosen part of the view). One may feel that the hypothesis of our result here looks very severe, but in fact the argument in the second last paragraph suggests that it would be significantly non-trivial to improve our result by relaxing the hypothesis.

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.,

[1] It is expected that such a security loss by using secure PRGs does not occur for any "single-party" cryptographic scheme where the user and the attacker are different. See Section 3.1 for a discussion in a typical case of public key encryption schemes.

[2] In the most strict formulation of two-party computation, the global parameter of the protocol such as the group $\mathbb{G}$ may have to be regarded as a part of the parties' inputs; in this case, the output of our PRG $\mathcal{R}^\dagger$ is regarded as dependent on a part of a party's input, which some reader may feel somewhat strange. However, in practice such a global parameter is often hardwired as a constant into an implementation of a protocol (e.g., by following a fixed standard choice of the "safe" group $\mathbb{G}$); now the output of our PRG $\mathcal{R}^\dagger$ does not depend on the variable inputs for the protocol.

[6, 7]) and the security issues caused by "backdoored PRGs" (e.g., [4, 5]). 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 [13, 14] 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 [14]) 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. [17]. 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* guaranteed in general that the resulting protocol with PRG is still secure. We note that Hazai and Zarosim in their recent paper [9] 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 [11], 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 PRG with close-to-ideal output distributions.

2 Preliminaries

2.1 Basic Notations and Settings

In this paper, we write $\{0,1\}^* = \bigcup_{n\geq 0}\{0,1\}^n$. We say that a function $\varepsilon(\lambda)$ of integers $\lambda \geq 1$ with $0 \leq \varepsilon(\lambda) \leq 1$ is *negligible* in λ, if for any polynomial poly with non-negative coefficients, there exists an integer $\lambda_0 \geq 1$ satisfying $\varepsilon(\lambda) < 1/\mathsf{poly}(\lambda)$ for any $\lambda > \lambda_0$.

For a probability distribution D, we write $a \leftarrow D$ to indicate that the element a is chosen according to the distribution D. Let $U(X)$ denote the uniform distribution on a set X, and let $a \xleftarrow{u} X$ mean $a \leftarrow U(X)$. We use a notation with 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, $[bb\colon b \xleftarrow{u} \{0,1\}]$ means $U(\{00, 11\})$. For two probability distributions X and Y over a common (finite) set Z, their *statistical distance* $\Delta(X,Y)$ is defined by

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

It is known that we have $\Delta(f(X), f(Y)) \leq \Delta(X,Y)$ for any probabilistic function f having its own internal randomness independent of X and Y.

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". When $\mathcal{A}$ is a non-uniform algorithm with auxiliary advice $z = z_\lambda$ (depending solely on the security parameter λ), we may either make the advice implicit in the notations, or write $\mathcal{A}^{(z_\lambda)}$ to clarify the given advice z_λ.

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 [8] with slight notational modifications.

Definition 1 (Indistinguishability)**.** Let $(I_\lambda)_{\lambda\geq 1}$ be a family of subsets $I_\lambda \subset \{0,1\}^*$ 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* (or *computationally close*) and write $X \stackrel{c}{\equiv} Y$, 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 $\lambda \geq 1$,

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

- We say that X and Y are *statistically indistinguishable* (or *statistically close*) and write $X \stackrel{s}{\equiv} Y$, if there exists a negligible function $\varepsilon(\lambda)$ satisfying $\Delta(X_{\lambda,w}, Y_{\lambda,w}) \leq \varepsilon(\lambda)$ for any $\lambda \geq 1$ and any $w \in I_\lambda$.

3 Security of Cryptography under Deviated Randomness

In this paper, we discuss the security of some kinds of cryptographic schemes in cases where its internal randomness is not ideal but slightly different from being ideal, more precisely, the distribution of the "deviated" internal randomness is (computationally or statistically) *indistinguishable from the uniform distribution*. We also consider the situation where the internal randomness may depend not only on a random seed to generate the randomness but also on some part of an input for the cryptographic scheme.

Informally, we say that a certain kind of security notion for a cryptographic scheme is *robust against computationally* (resp. *statistically*) *deviated randomness*, if the scheme is still secure even when the ideal internal randomness for the scheme is replaced by any randomness that is computationally (resp. statistically) indistinguishable from being ideal. Our main target for the notion in this paper is the case of secure two-party computation in the semi-honest model; for the case, we give a more concrete formulation of the robustness against deviated randomness in Section 3.2. On the other hand, one may feel that the robustness of security against deviated randomness should be trivial for "ordinary" situations in cryptography provided the deviated internal randomness is indistinguishable from being ideal. The author indeed supports the intuition for the case of "single-party" cryptographic schemes; as a supporting evidence, in Section 3.1 we study the robustness against deviated randomness in the case of public key encryption as a typical example of such cryptographic schemes.

3.1 Public Key Encryption under Deviated Randomness

As a toy example of "ordinary" situations in cryptography, here we formalize the robustness of the CPA security (security against chosen-plaintext attacks) for public key encryption (PKE) schemes against deviated randomness in the aforementioned sense. First we clarify the definition of secure pseudorandom generators in this paper, which is slightly modified for enabling to deal with the case where the output distribution may depend on some part of an input for a cryptographic scheme.

Definition 2 (Pseudorandom generators)**.** Let $\lambda \geq 1$ be a security parameter, and let $(I_\lambda)_{\lambda \geq 1}$ be a family of subsets $I_\lambda \subset \{0,1\}^*$ indexed by λ. We call an algorithm $\mathcal{R} = \mathcal{R}(1^\lambda, w; s)$, with random *seed* s, security parameter λ and additional input $w \in I_\lambda$, a *pseudorandom generator* (*PRG*) parameterized by elements of I_λ; and we say that $\mathcal{R}$ is PPT, if it is PPT with respect to the security parameter λ. Then we say that $\mathcal{R}$ is *computationally* (resp. *statistically*) *secure*, if the family $\big(\mathcal{R}(1^\lambda, w; s)\big)_{\lambda, w}$ of the output distributions with uniformly random seeds s is computationally (resp. statistically) indistinguishable from the uniform distribution (in the sense of Definition 1).

Similarly to the ordinary definition, a PKE scheme with deviated randomness for encryption can be formalized as a collection of the following algorithms; a PPT *key generation algorithm* $\mathsf{Gen}(1^\lambda)$ with security parameter λ outputs a pair of a public key pk and the corresponding secret key sk; a PPT *encryption algorithm* $\mathsf{Enc}(\mathsf{pk}, m; \mathcal{R}(1^\lambda, \mathsf{pk}, m))$ outputs a ciphertext for a plaintext m, where $\mathcal{R}$ is a PRG for generating the internal randomness parameterized by pk and m; and a polynomial-time *decryption algorithm* $\mathsf{Dec}(\mathsf{sk}, c)$ outputs a plaintext for a ciphertext c. The ordinary syntax for a PKE scheme corresponds to the case where $\mathcal{R}$ is ideally random, that is, the output of $\mathcal{R}(1^\lambda, \mathsf{pk}, m)$ for a given security parameter λ is uniformly random and independent of pk and m. The attack success probability $\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda)$ of a (non-uniform) adversary $\mathcal{A}$ against the CPA security with security parameter λ under the use of the PRG $\mathcal{R}$ is given by

$$\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda) = \Pr\left[b = b^* \colon \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda);\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{submit}, 1^\lambda, \mathsf{pk});\ b^* \xleftarrow{u} \{0,1\}; \\ c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{b^*}; \mathcal{R}(1^\lambda, \mathsf{pk}, m_{b^*}));\ b \leftarrow \mathcal{A}(\mathsf{guess}, 1^\lambda, \mathsf{st}, \mathsf{pk}, c^*) \end{array}\right].$$

We say that a PKE scheme is *CPA secure under the PRG* $\mathcal{R}$ *against non-uniform adversaries*, if for any PPT non-uniform adversary $\mathcal{A}$, $\left|\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda) - 1/2\right|$ is negligible in λ. We may omit the $\mathcal{R}$ in the notations and terminology when $\mathcal{R}$ is ideally random. Then the following fact means that the CPA security of PKE schemes against non-uniform adversaries is robust against computationally deviated randomness:

Proposition 1. *If a PKE scheme is CPA secure against non-uniform adversaries, then for any PPT and computationally secure PRG* $\mathcal{R} = \mathcal{R}(1^\lambda, \mathsf{pk}, m)$ *parameterized by a public key* pk *and a plaintext* m*, the PKE scheme is also CPA secure under the PRG* $\mathcal{R}$ *against non-uniform adversaries.*

Proof. The proof is basically an ordinary game-hopping argument that replaces the PRG $\mathcal{R}$ with an ideally random variable owing to the computational security of $\mathcal{R}$, though the description of the argument is somewhat complicated for dealing with non-uniform adversaries. A full proof is given in Appendix A. □

3.2 Secure Two-Party Computation under Deviated Randomness

Here we formalize the robustness of the security for two-party computation protocols in the semi-honest model against deviated randomness in the aforementioned sense. 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 [8] and in [12].

3.2.1 Basic Notations and Terminology

We summarize some notations and terminology for two-party protocols. Let π be a two-party protocol between parties $\mathcal{P}_1$ and $\mathcal{P}_2$. Formally, the parties are modeled as interactive probabilistic 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, the internal random tape for $\mathcal{P}_i$ at an execution of π is denoted by r_i, and the output obtained by $\mathcal{P}_i$ after an execution of π is denoted by $\mathsf{out}^\pi_i(1^\lambda, x_1, x_2; r_1, r_2)$, or by $\mathsf{out}^\pi_i(1^\lambda, \vec{x}; \vec{r})$ 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 $\mathsf{out}^\pi_1(1^\lambda, \vec{x}; \vec{r})$ and $\mathsf{out}^\pi_2(1^\lambda, \vec{x}; \vec{r})$ in this order. In this paper, we suppose (unless otherwise specified) that π has polynomial (in λ) time and communication complexity. Accordingly, we assume that the inputs x_1, x_2 are elements of $\{0,1\}^*$ and their lengths are bounded by a polynomial in

λ. We also assume that $r_1 \in \{0,1\}^{\rho_1(\lambda)}$ and $r_2 \in \{0,1\}^{\rho_2(\lambda)}$ for some polynomials $\rho_1 = \rho_1^\pi$ and $\rho_2 = \rho_2^\pi$. 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 $\mathcal{P}_i$ (as well as the security parameter 1^λ).

We let a *functionality* with input set $I \subset \{0,1\}^* \times \{0,1\}^*$ 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. In this paper, we suppose that f is polynomial-time computable; more precisely, there exists an algorithm that runs within polynomial time in $|\vec{x}|$ and computes the values of $f_1(\vec{x})$ and $f_2(\vec{x})$ from $\vec{x}$ and the internal randomness for f_1 and f_2. 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.

3.2.2 Definitions of Security and Robustness under Deviated Randomness

Here we describe the definition of security for a two-party protocol π in the semi-honest model[3]. 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)**.** For $i \in \{1,2\}$, we say that a two-party protocol π *securely computes a functionality* $f = (f_1, f_2)$ *against semi-honest Party* $\mathcal{P}_i$, if there exists a PPT algorithm $\mathcal{S}_i$, called a *simulator* for $\mathcal{P}_i$, satisfying

$$\begin{aligned} &\big(\mathcal{S}_i(1^\lambda, x_i, f_i(\vec{x})), f(\vec{x})\big)_{\lambda,\vec{x}} \\ &\stackrel{c}{\equiv} \big(\big[x_i, r_i, \mathsf{trans}_i^\pi(1^\lambda, \vec{x}; \vec{r}), \mathsf{out}^\pi(1^\lambda, \vec{x}; \vec{r})\colon r_1 \stackrel{u}{\leftarrow} \{0,1\}^{\rho_1(\lambda)}, r_2 \stackrel{u}{\leftarrow} \{0,1\}^{\rho_2(\lambda)}\big]\big)_{\lambda,\vec{x}} \end{aligned} \tag{1}$$

where the indices λ and $\vec{x}$ run over the ranges $\lambda \geq 1$ and $\vec{x} \in X_\lambda$. In this case, we say that this simulator $\mathcal{S}_i$ is *computationally indistinguishable*. On the other hand, we say that a simulator $\mathcal{S}_i$ is *statistically indistinguishable*, if the random variable families at the two sides of (1) are statistically indistinguishable. Moreover, we say that π *securely computes* $f = (f_1, f_2)$ *in the semi-honest model*, if π securely computes f against semi-honest $\mathcal{P}_i$ for each $i \in \{1,2\}$.

From now, we consider an extension of the definition above to cases where one of the two parties uses a PRG (endowed with its own random seed) for generating the party's random tape. We put a technical assumption that a PRG $\mathcal{R}$ used by Party $\mathcal{P}_i$ ($i \in \{1,2\}$) has internal random seed s of length $\widetilde{\rho}_i(\lambda)$ for some polynomial $\widetilde{\rho}_i$ and is parameterized by an input x_i of the party, i.e., it is of the form $\mathcal{R}(1^\lambda, x_i; s)$ with $s \stackrel{u}{\leftarrow} \{0,1\}^{\widetilde{\rho}_i(\lambda)}$; and its output has precisely length $\rho_i(\lambda)$. Then we regard "the execution of π where Party $\mathcal{P}_i$ uses a PRG $\mathcal{R}$" as an individual two-party protocol, which is denoted by $\pi \circ_i \mathcal{R}$ and defined as follows:

- The input set for $\pi \circ_i \mathcal{R}$ is the same as that for 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 $\{0,1\}^{\widetilde{\rho}_i(\lambda)}$ 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 $s_i \stackrel{u}{\leftarrow} \{0,1\}^{\widetilde{\rho}_i(\lambda)}$ and $r_{3-i} \stackrel{u}{\leftarrow} \{0,1\}^{\rho_{3-i}(\lambda)}$ for Parties $\mathcal{P}_i$ and $\mathcal{P}_{3-i}$, respectively, to execute the protocol $\pi \circ_i \mathcal{R}$, first $\mathcal{P}_i$

[3]The notion is often called with different names in the literature; e.g., "π privately computes f" in Section 7.2 of [8]; and "π securely computes f in the presence of static semi-honest adversaries" in [12].

runs $\mathcal{R}(1^\lambda, x_i; s_i)$ with random 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; but we note also that the r_i used during the protocol execution can be deterministically recovered from s_i and the party's input, both of which are included in the view for the party in the new protocol $\pi \circ_i \mathcal{R}$.

Then we formalize the robustness of the security for two-party protocols in the semi-honest model against deviated randomness as follows:

Definition 4 (Robustness of security against deviated randomness)**.** Let $i \in \{1,2\}$, and suppose that a two-party protocol π securely computes $f = (f_1, f_2)$ against semi-honest Party $\mathcal{P}_i$. Then we say that the security of π against semi-honest Party $\mathcal{P}_i$ is *robust against computationally* (resp. *statistically*) *deviated randomness*, if the protocol $\pi \circ_i \mathcal{R}$ defined above securely computes f against semi-honest $\mathcal{P}_i$ for any PPT and computationally (resp. statistically) secure PRG $\mathcal{R}$ for Party $\mathcal{P}_i$.

4 *Non*-Robust Two-Party Protocol against Deviated Randomness

In Section 3, we introduced a notion concerning (non-)preservation of some security property for a certain cryptographic scheme in cases where the ideal internal randomness for the scheme is replaced with an output of a cryptographically secure PRG. We also verified as a toy example that the CPA security of PKE schemes is indeed preserved by the use of such a PRG as we naturally expect. In contrast, in this section we show that the (computational) security of two-party protocols in the semi-honest model is NOT ALWAYS preserved by the use of such a PRG, even if the PRG has significantly stronger *statistical* security. We emphasize that a two-party protocol that provides a counterexample here is found even from the existing protocols in the literature, rather than those artificially designed to prove our theorem. We also note that the output distribution of the PRG in our counterexample depends on publicly known global parameters for the protocol but not on an individual input for each party.

4.1 The Protocol

As a two-party protocol that provides our counterexample, here we use an oblivious transfer protocol proposed by Asharov et al. in ACM CCS 2013 [3]. We refer to Protocol 51 in Section 5.2 of the full version for the paper [3] for the description of the protocol. 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 chosen from some sets of bit strings. Our description of the protocol is summarized in Figure 1; we explain the details of the construction below. Before starting the detailed explanation, we emphasize that the inputs for the parties in the protocol are divided into a set of publicly known global parameters that are reused in successive executions of the protocol, and "actual" local inputs for the parties at each individual execution of the protocol[4]. Moreover, the output distribution of the PRG for our counterexample constructed in Section 4.2 below depends on the global parameter but not on an individual input.

The global parameter for the protocol involves mainly a cyclic group $\mathbb{G}$ (written in multiplicative form) of publicly known order q generated by an element g, and a polynomial-time key derivation function KDF that generates an ℓ-bit key string (for a certain parameter ℓ) from a given element of $\mathbb{G}$. A basic assumption on the group $\mathbb{G}$ here is the decisional Diffie–Hellman (DDH) assumption, i.e., given the $\mathbb{G}$, q, g and a uniformly random element $h \in \mathbb{G}$, the distributions of (g, h, g^a, h^a) and (g, h, g^a, g^b) with $a, b \xleftarrow{u} \{0, \ldots, q-1\}$ are computationally indistinguishable. Moreover, in fact there is a significantly stronger requirement for the

[4]In fact, in the original paper [3], the protocol was designed in a way that n executions of the oblivious transfer can be performed at once by using the same global parameter. In the present paper, we only focus on the simplest case of a single oblivious transfer (i.e., $n = 1$) to simplify the argument.

Public global parameters: t-bit prime p, cyclic subgroup $\mathbb{G} = \langle g \rangle$ of $(\mathbb{F}_p)^\times$, $q = \lvert\mathbb{G}\rvert$, and a key derivation function $\mathsf{KDF}\colon \mathbb{G} \to \{0,1\}^\ell$		
(Party $\mathcal{P}_1$ (Sender))	Input: $(x^{(0)}, x^{(1)}) \in \{0,1\}^\ell$	Output: (none)
	Random tape: $r_1 \in \{0,1\}^{2t}$	
(Party $\mathcal{P}_2$ (Receiver))	Input: $\sigma \in \{0,1\}$	Output: $x^{(\sigma)}$
	Random tape: $(r'_2, r''_2) \in \{0,1\}^{2t} \times \{0,1\}^{2t}$	

1. **(Computation of $\mathcal{H}_p(r'_2)$)** Receiver computes $h' \leftarrow r'_2 \bmod p$, and if $h'^{(p-1)/q} \bmod p \notin \{0,1\}$ then sets $h \leftarrow h'^{(p-1)/q} \bmod p$, or else sets $h \leftarrow 1$.
2. Receiver computes $\alpha \leftarrow r''_2 \bmod q$, and sets $(h^{(0)}, h^{(1)}) \leftarrow (g^\alpha, h)$ if $\sigma = 0$ and $(h^{(0)}, h^{(1)}) \leftarrow (h, g^\alpha)$ if $\sigma = 1$ (computed in the group $\mathbb{G}$). Then Receiver sends $(h^{(0)}, h^{(1)})$ to Sender.
3. Sender computes $r \leftarrow r_1 \bmod q$, $u \leftarrow g^r \in \mathbb{G}$, $(k^{(0)}, k^{(1)}) \leftarrow ((h^{(0)})^r, (h^{(1)})^r)$ 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 $v^{(\sigma)} \oplus \mathsf{KDF}(u^\alpha)$; while Sender outputs nothing.

Figure 1: An oblivious transfer protocol from [3] for our counterexample (here we assume that $t \geq \lambda$; and $\oplus$ denotes bit-wise XOR)

group, whose explanation is quoted from the text in the second paragraph of Section 5.2 in the full version of [3] as follows (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 $h \neq 1$.* [.....]

Following the quoted specification, we use the subgroup of order q of the multiplicative group $(\mathbb{F}_p)^\times$ for the finite field $\mathbb{F}_p$ (denoted by $\mathbb{Z}_p$ in the quoted text) of prime order p (where q divides $p-1$) as the underlying group $\mathbb{G}$,[5] and we adopt the following algorithm (modified in order to guarantee that it halts within finite steps) to sample a (statistically close to) uniformly random element h of $\mathbb{G}$: Choose $x \xleftarrow{u} \mathbb{F}_p$, and if $h = x^{(p-1)/q} \notin \{0,1\}$ then output the h, or else output $1 \in \mathbb{G}$. We note that, by setting the parameters p and q exponentially large with respect to λ, the probability that $h \in \{0,1\}$ in the process above, which is $(1 + (p-1)/q)/p \approx 1/p + 1/q$, is exponentially small and hence the probability distribution of the element of $\mathbb{G}$ chosen as above is indeed statistically close to uniform.

Moreover, in order to generate a random element x of $\mathbb{F}_p$ by using a random bit sequence (which is suitable for our formulation in Section 3.2.1) rather than direct sampling over $\mathbb{F}_p$, here we use the following well-known technique: For a t-bit prime p, choose $x' \xleftarrow{u} \{0,1\}^{2t}$, identify the x' with the binary expression of an integer in $\{0, \ldots, 2^{2t} - 1\}$, and set $x \leftarrow x' \bmod p \in \mathbb{F}_p$. Now the statistical distance between $U(\mathbb{F}_p)$ and the distribution of the x is exponentially small with respect to λ if we set e.g., $t \geq \lambda$, which is implied by the following lemma:

Lemma 1. *Let M, N be two positive integers. Put $\delta = M/N - \lfloor M/N \rfloor$, hence in particular $0 \leq \delta < 1$. 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 $\Pr[U_M \bmod N = a] = (\lfloor M/N \rfloor + 1)/M > 1/N$ for $0 \leq a \leq M - \lfloor M/N \rfloor N - 1$

[5] The authors of [3] in fact also proposed to use elliptic curve groups. This would be more efficient than the current subgroup from finite fields, but the very efficiency is not relevant to our argument in the present paper.

and $\Pr[U_M \bmod N = a] = \lfloor M/N \rfloor / M \leq 1/N$ for $M - \lfloor M/N \rfloor N \leq a \leq N-1$. 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. □

By combining these arguments, we define an algorithm $\mathcal{H}_p$ as follows:

- Given an input $x' \in \{0,1\}^{2t}$, compute $x \leftarrow x' \bmod p \in \mathbb{F}_p$, and if $x^{(p-1)/q} \notin \{0,1\}$ then output $x^{(p-1)/q}$, or else output $1 \in \mathbb{G}$.

Then the output distribution of $\mathcal{H}_p(x')$ with $x' \xleftarrow{u} \{0,1\}^{2t}$ is statistically indistinguishable from $U(\mathbb{G})$.

On the other hand, for the (deterministic) key derivation function $\mathsf{KDF}\colon \mathbb{G} \to \{0,1\}^\ell$, the requirement here is the following: The two distributions of $(\mathbb{G}, q, g, g^r, x', \mathsf{KDF}(\mathcal{H}_p(x')^r))$ and of $(\mathbb{G}, q, g, g^r, x', z)$ with $r \xleftarrow{u} \{0, \ldots, q-1\}$, $x' \xleftarrow{u} \{0,1\}^{2t}$ and $z \xleftarrow{u} \{0,1\}^\ell$ are computationally indistinguishable; cf., Appendix A of the full version for [3]. We note that a concrete construction of such KDF was not discussed even in the original paper [3]; accordingly, in the present paper we also just assume that such KDF exists.

Summarizing, we obtain the protocol description in Figure 1. We note that the protocol correctly computes the desired output of Receiver, since $u^\alpha = (g^r)^\alpha = (g^\alpha)^r = (h^{(\sigma)})^r = k^{(\sigma)}$ and hence $v^{(\sigma)} \oplus \mathsf{KDF}(u^\alpha) = v^{(\sigma)} \oplus \mathsf{KDF}(k^{(\sigma)}) = x^{(\sigma)}$. On the other hand, here we only focus on the security against semi-honest Receiver which is sufficient for our purpose, though it is also secure against semi-honest Sender. We give a security proof for the sake of completeness:

Proposition 2. *The protocol in Figure 1 securely computes the specified functionality against semi-honest Receiver (Party $\mathcal{P}_2$) under the assumptions above on the choices of $\mathbb{G}$ and KDF.*

Proof. First, the real view for Receiver in the protocol consists of $u \in \mathbb{G}$ and $v^{(0)}, v^{(1)} \in \{0,1\}^\ell$ as well as the local input σ, local output $x^{(\sigma)}$, and the random tape (r_2', r_2''). To construct a simulator $\mathcal{S}_2 = \mathcal{S}_2(1^\lambda, \sigma, x^{(\sigma)})$ for the Receiver's view, we focus only on the case $\sigma = 0$ as the other case $\sigma = 1$ is similar by symmetry.

The simulator $\mathcal{S}_2(1^\lambda, 0, x^{(0)})$ is defined to first choose a uniformly random tape (r_2', r_2'') for Receiver and compute the elements $h \in \mathbb{G}$, $\alpha \in \{0, \ldots, q-1\}$, and $h^{(0)} = g^\alpha$ in the same manner as the real protocol. Then the simulator randomly chooses $r \in \{0, \ldots, q-1\}$ and computes elements $u \leftarrow g^r$ and $k^{(0)} \leftarrow (h^{(0)})^r$ as Sender does in the real protocol. Moreover, the simulator computes $v^{(0)} \leftarrow x^{(0)} \oplus \mathsf{KDF}(k^{(0)})$ and chooses $\overline{v}^{(1)} \xleftarrow{u} \{0,1\}^\ell$. The simulator outputs (r_2', r_2'') as the simulated random tape and $(u, v^{(0)}, \overline{v}^{(1)})$ as the simulated transcript.

To prove that the simulator's output distribution is computationally indistinguishable from the distribution of a real view, first note that the components r_2', r_2'', u, and $v^{(0)}$ are perfectly simulated by the construction. On the other hand, since the choice of a sufficiently long random tape r_1 implies that the distribution of $r = r_1 \bmod q$ is statistically close to uniform (cf. Lemma 1), we may assume that $r \xleftarrow{u} \{0, \ldots, q-1\}$ in both the simulator and the real execution of the protocol.

Note that $v^{(1)} = x^{(1)} \oplus \mathsf{KDF}(k^{(1)}) = x^{(1)} \oplus \mathsf{KDF}(h^r) = x^{(1)} \oplus \mathsf{KDF}(\mathcal{H}_p(r_2')^r)$ in the real protocol. Now the assumption on the choice of KDF implies that the distribution of $\mathsf{KDF}(\mathcal{H}_p(r_2')^r)$, hence the distribution of $x^{(1)} \oplus \mathsf{KDF}(\mathcal{H}_p(r_2')^r)$ as well, is computationally close to $U(\{0,1\}^\ell)$ even when the objects $\mathbb{G}$, q, g, $g^r = u$ and r_2' are seen. This implies that the component $\overline{v}^{(1)}$ of the simulator's output also simulates the component $v^{(1)}$ of the real view in a computationally indistinguishable manner. This completes the proof. □

4.2 The PRG

To provide our counterexample, here we construct a PRG $\mathcal{R}^\dagger$ which is even statistically secure but for which the two-party protocol in Section 4.1 denoted here by $\pi^\dagger$ becomes not secure against semi-honest Receiver

(Party $\mathcal{P}_2$) when Receiver uses the PRG $\mathcal{R}^\dagger$ to generate the internal random tape, i.e., the protocol $\pi^\dagger \circ_2 \mathcal{R}^\dagger$ is not secure against semi-honest Party $\mathcal{P}_2$. Hence, it gives an example of a secure two-party protocol whose security is not robust against even statistically deviated randomness.

Recall that, in the construction of the protocol $\pi^\dagger$, the function $\mathcal{H}_p$ enables Receiver to sample a random element h of the group $\mathbb{G} = \langle g \rangle$ without knowing its discrete logarithm with respect to g even if Receiver can see the internal randomness used to sample the h. Intuitively, our construction of the PRG is intended to disable the hiding property of the function $\mathcal{H}_p$. Before the construction, we prepare a lemma:

Lemma 2. *Let L, M, N be positive integers with $L \leq N$. 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*

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

Proof. We have $a \geq 0$ and $a - L < 0$ by the choice of a. On the other hand, since $(N-1-a)/L < K \leq (N-1-a)/L + 1$, we have $a + (K-1) \cdot L \leq N-1$ 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 1. □

Now we define the PRG $\mathcal{R}^\dagger$ with random seed $s = (s_1, s_2, s_3, s_4) \in \{0,1\}^{2t} \times \{0,1\}^{2t} \times \{0,1\}^{3t} \times \{0,1\}^{2t}$, parameterized by global parameters $\mathbb{G}$, p, q, and g for the protocol $\pi^\dagger$ (but not dependent on Receiver's local input σ). Here we assume that $(p-1)/q$ is coprime to q; this may be often satisfied, e.g., when p is a safe prime $p = 2q+1$. The construction of $\mathcal{R}^\dagger$ is described as follows, where g_0 is a fixed generator of $(\mathbb{F}_p)^\times$ and $d \in \{1, \ldots, q-1\}$ is the multiplicative inverse of $(p-1)/q$ modulo q:

1. Compute $e \leftarrow s_1 \bmod q$ and $h^\dagger \leftarrow g^e \in \mathbb{G}$.
2. Compute $e' \leftarrow s_2 \bmod (p-1)$ and $h^{\dagger\dagger} \leftarrow (h^\dagger)^d \cdot {g_0}^{qe'} \in (\mathbb{F}_p)^\times$.
3. Compute $r^\dagger \leftarrow h^{\dagger\dagger} + (s_3 \bmod K) \cdot p \in \{0, \ldots, 2^{2t}-1\}$ where $K = \lfloor (2^{2t} - 1 - h^{\dagger\dagger})/p \rfloor + 1$, identify $r^\dagger$ with a $2t$-bit sequence, and then output the pair $(r^\dagger, s_4)$ of the $r^\dagger \in \{0,1\}^{2t}$ and s_4.

Proposition 3. *For a uniformly random seed $s = (s_1, s_2, s_3, s_4)$, the output distribution of $\mathcal{R}^\dagger(s)$ is statistically close to $U(\{0,1\}^{2t} \times \{0,1\}^{2t})$, and $\mathcal{H}_p(r^\dagger) = g^e$ where e and $r^\dagger$ are as in the construction of $\mathcal{R}^\dagger$.*

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

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

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

For the first assertion, it suffices to show that $r^\dagger \stackrel{s}{\equiv} U(\{0,1\}^{2t})$. Let $f \in \{0, \ldots, p-2\}$ be the discrete logarithm of $g \in \mathbb{G} \subset (\mathbb{F}_p)^\times$ with respect to g_0. Then f divides $(p-1)/q$ since $g^q = 1$; we put $f = f'(p-1)/q$ with $1 \leq f' \leq q-1$. Now 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 1, we may assume without loss of generality that $e \stackrel{u}{\leftarrow} \{0, \ldots, q-1\}$ and $e' \stackrel{u}{\leftarrow} \{0, \ldots, p-2\}$. Now we have $h^{\dagger\dagger} = g^{ed} \cdot {g_0}^{qe'} = {g_0}^{fed + qe'}$. Since $fed + qe' \bmod q = e \cdot fd \bmod q$ and fd is coprime to q by the argument above, it follows from the choice of e that $fed + qe' \bmod q$ is uniformly random. On the other hand, since $\lfloor (fed + qe')/q \rfloor = e' + \lfloor fed/q \rfloor$, it follows from the choice of e' that $\lfloor (fed + qe')/q \rfloor \bmod (p-1)/q$ is uniformly random and is independent of $fed + qe' \bmod q$. 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 2 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 $r'_2 \xleftarrow{u} \{0,1\}^{2t}$, then the distribution of $r^\dagger$ would be statistically close to $U(\{0,1\}^{2t})$ by the argument above. Moreover, since p is exponentially large and $r'_2 \in \{0,1\}^{2t}$ is sufficiently long, Lemma 2 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 are statistically indistinguishable. By these arguments, it follows that the distribution of $r^\dagger$ is in fact statistically close to $U(\{0,1\}^{2t})$. This completes the proof. □

By Proposition 3, $\mathcal{R}^\dagger$ is a statistically secure PRG for the internal random tape for Receiver in the protocol $\pi^\dagger$. On the other hand, the protocol $\pi^\dagger \circ_2 \mathcal{R}^\dagger$ is never secure against semi-honest Receiver; indeed, by using the output of $\mathcal{R}^\dagger$ instead of the original random tape (r'_2, r''_2), Proposition 3 implies that Receiver can know the exponent of $h = \mathcal{H}_p(r^\dagger) = g^e$ by using the seed s for the PRG as $e = s_1 \bmod q$, therefore Receiver can recover the other input $x^{(1-\sigma)}$ of Sender from the messages u and $v^{(1-\sigma)}$ sent from Sender as $x^{(1-\sigma)} = v^{(1-\sigma)} \oplus \mathsf{KDF}(u^e)$, since $k^{(1-\sigma)} = (h^{(1-\sigma)})^r = h^r = (g^e)^r = (g^r)^e = u^e$. This yields the following theorem, which is the main result of the present paper:

Theorem 1. *The security of a general two-party protocol against a semi-honest party is NOT robust against even* statistically *deviated randomness, namely, there are a two-party protocol $\pi^\dagger$ and a statistically secure PRG $\mathcal{R}^\dagger$ satisfying the following: $\pi^\dagger$ securely computes a functionality in the semi-honest model, but $\pi^\dagger \circ_2 \mathcal{R}^\dagger$ is not secure against semi-honest Party $\mathcal{P}_2$.*

5 How to Prove Robustness against Deviated Randomness

In Section 4, we showed that the security of a two-party protocol in the semi-honest model is in general *not* robust against (even statistically) deviated randomness. In contrast, this section is devoted to try to develop some affirmative result on proving the robustness of the security for two-party protocols against deviated randomness under certain conditions.

In order to establish the affirmative result on the robustness against deviated randomness, we propose the following notion regarding simulators for two-party protocols:

Definition 5. We say that a simulator $\mathcal{S}_i$ for a view of 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 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 $\mathcal{P}_i$, and then runs $\mathcal{T}_i(1^\lambda, x_i, f_i(\vec{x}), r_i)$ to obtain a simulated transcript trans_i for $\mathcal{P}_i$ and outputs $(x_i, r_i, \mathsf{trans}_i)$.

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 (as is frequently done in a security proof for a two-party protocol). For example, the simulator $\mathcal{S}_2$ constructed in the proof of Proposition 2 is in fact with raw random tape in this sense. Then we give the following result:

Theorem 2. *Let π be any two-party protocol, and let f be any functionality. For each $i \in \{1,2\}$, if π securely computes f against semi-honest $\mathcal{P}_i$ with* statistically indistinguishable *simulator* with raw random tape *in the sense of Definition 5, then the security of π against semi-honest $\mathcal{P}_i$ is robust against* statistically *deviated randomness. Moreover, for any PPT and statistically secure PRG $\mathcal{R}$ for the random tape of $\mathcal{P}_i$, the protocol $\pi \circ_i \mathcal{R}$ also securely computes f against semi-honest $\mathcal{P}_i$ with statistically indistinguishable simulator with raw random tape.*

Proof. Here we focus on the case $i = 1$ only, as the other case $i = 2$ is similar by symmetry. By the assumption, there exists a PPT statistically indistinguishable simulator $\mathcal{S}_1$ with raw random tape for $\mathcal{P}_1$ in π. Let $\mathcal{T}_1$ denote the PPT algorithm yielded by the "raw random tape" condition as in Definition 5. By definition, $(\mathcal{X}_{\lambda,\vec{x}})_{\lambda,\vec{x}} \stackrel{\text{def}}{=} \big(x_1, r_1, \mathcal{T}_1(1^\lambda, x_1, f_1(\vec{x}), r_1), f(\vec{x})\big)_{\lambda,\vec{x}}$ is statistically indistinguishable from $(\mathcal{Y}_{\lambda,\vec{x}})_{\lambda,\vec{x}} \stackrel{\text{def}}{=} \big(x_1, r_1, \mathsf{trans}_1^\pi(1^\lambda, \vec{x}; r_1, r_2), \mathsf{out}^\pi(1^\lambda, \vec{x}; r_1, r_2)\big)_{\lambda,\vec{x}}$, where r_1 and r_2 are uniformly random for both $\mathcal{X}_{\lambda,\vec{x}}$ and $\mathcal{Y}_{\lambda,\vec{x}}$.

First, let $\overline{r}_1 = \overline{r}_1(1^\lambda, x_1)$ denote the random variable identical to the output distribution of $\mathcal{R}(1^\lambda, x_1)$. Then we have $\overline{r}_1 \stackrel{s}{\equiv} r_1$ by the assumption on $\mathcal{R}$. Therefore, the family of $\mathcal{X}'_{\lambda,\vec{x}}$ obtained by replacing each r_1 appeared in $\mathcal{X}_{\lambda,\vec{x}}$ with $\overline{r}_1$ is also statistically indistinguishable from the family of $\mathcal{Y}'_{\lambda,\vec{x}}$ obtained by replacing each r_1 appeared in $\mathcal{Y}_{\lambda,\vec{x}}$ with $\overline{r}_1$.

Secondly, for each possible value of $\overline{r}_1$, let $s(\overline{r}_1) = s(\overline{r}_1(1^\lambda, x_1))$ denote the random variable that follows the conditional distribution of the random seed $\widetilde{r}_1$ for $\mathcal{R}(1^\lambda, x_1)$ conditioned on the event $\mathcal{R}(1^\lambda, x_1; \widetilde{r}_1) = \overline{r}_1$. Then, by applying to both $\mathcal{X}'_{\lambda,\vec{x}}$ and $\mathcal{Y}'_{\lambda,\vec{x}}$ the common probabilistic function that replaces $\overline{r}_1$ in the second component of the distribution with a random value of $s(\overline{r}_1)$, it follows that the two families of the resulting distributions $\mathcal{X}''_{\lambda,\vec{x}}$ and $\mathcal{Y}''_{\lambda,\vec{x}}$, respectively, are also statistically indistinguishable.

Now since $\overline{r}_1$ follows the output distribution of $\mathcal{R}(1^\lambda, x_1; \widetilde{r}_1)$ with uniformly random $\widetilde{r}_1$ by definition, it follows that the distribution of $s = s(\overline{r}_1(1^\lambda, x_1; \widetilde{r}_1))$ with uniformly random $\widetilde{r}_1$ (and with the own internal randomness for s) is identical to the uniform distribution of the random seed for $\mathcal{R}(1^\lambda, x_1)$. Moreover, we always have $\mathcal{R}(1^\lambda, x_1; s(\overline{r}_1)) = \overline{r}_1$ by the definition of $s(\overline{r}_1)$. Therefore, the random variable $\mathcal{X}''_{\lambda,\vec{x}}$ can be rewritten as

$$\mathcal{X}''_{\lambda,\vec{x}} = \big(x_1, s, \mathcal{T}_1(1^\lambda, x_1, f_1(\vec{x}), \mathcal{R}(1^\lambda, x_1; s)), f(\vec{x})\big)$$

where the s follows the uniform distribution for the random seed for $\mathcal{R}(1^\lambda, x_1)$, and the random variable $\mathcal{Y}''_{\lambda,\vec{x}}$ can be rewritten as

$$\mathcal{Y}''_{\lambda,\vec{x}} = \big(x_1, s, \mathsf{trans}_1^\pi(1^\lambda, \vec{x}; \mathcal{R}(1^\lambda, x_1; s), r_2), \mathsf{out}^\pi(1^\lambda, \vec{x}; \mathcal{R}(1^\lambda, x_1; s), r_2)\big)$$

where the s follows again the uniform distribution for the random seed for $\mathcal{R}(1^\lambda, x_1)$. By the definition of $\pi \circ_1 \mathcal{R}$, this $\mathcal{Y}''_{\lambda,\vec{x}}$ is also equal to

$$\mathcal{Y}''_{\lambda,\vec{x}} = \big(x_1, s, \mathsf{trans}_1^{\pi\circ_1\mathcal{R}}(1^\lambda, \vec{x}; s, r_2), \mathsf{out}^{\pi\circ_1\mathcal{R}}(1^\lambda, \vec{x}; s, r_2)\big) \ .$$

According to the argument above, we now define a simulator $\widetilde{S}_1$ for $\mathcal{P}_1$ in $\pi \circ_1 \mathcal{R}$ with raw random tape as follows: Given 1^λ, x_1 and $f_1(\vec{x})$ as inputs, $\widetilde{\mathcal{S}}_1$ generates s uniformly at random, computes $\mathsf{trans}_1 \leftarrow \mathcal{T}_1(1^\lambda, x_1, f_1(\vec{x}), \mathcal{R}(1^\lambda, x_1; s))$, and then outputs $(x_1, s, \mathsf{trans}_1)$. This is a PPT algorithm as well as $\mathcal{T}_1$ and $\mathcal{R}$, and now $\mathcal{X}''_{\lambda,\vec{x}}$ is identical to $\big(\widetilde{\mathcal{S}}_1(1^\lambda, x_1, f_1(\vec{x})), f(\vec{x})\big)$. Therefore, by the argument above, it follows that the simulator $\widetilde{\mathcal{S}}_1$ for $\mathcal{P}_1$ in $\pi \circ_1 \mathcal{R}$ is statistically indistinguishable. This completes the proof. □

Here we give a remark on the three conditions in the statement above: "statistical" indistinguishability of the simulator; "raw random tape" property of the simulator; and "statistical" security of the PRG. Among them, the first condition on the statistical indistinguishability of the simulator cannot be relaxed to computational indistinguishability, as the counterexample given in Section 4 satisfies the other two conditions in the statement but now the conclusion fails. On the other hand, the second condition on the raw random tape property of the simulator cannot be removed as well, since there exists an example of a secure two-party protocol and a PRG that satisfies the other two conditions in the statement but now the conclusion fails; see Appendix B for a concrete construction of such an example.

One may feel that, among the three conditions above, the statistical security of the PRG is most stressful; one would expect that, in order to achieve only computational (rather than unconditional) security, any "secure" protocol should allow a party to use any computationally secure PRG to prepare the party's own random tape. Unfortunately, the use of such a PRG causes the following hurdle in our proof strategy above. We recall that, in the proof, we had the relation $\mathcal{X}'_{\lambda,\vec{x}} \stackrel{s}{\equiv} \mathcal{Y}'_{\lambda,\vec{x}}$ owing to the statistical security of the PRG. When the PRG is just computationally secure, the relation becomes to a weaker relation $\mathcal{X}'_{\lambda,\vec{x}} \stackrel{c}{\equiv} \mathcal{Y}'_{\lambda,\vec{x}}$. Now *if the random variable $s(\overline{r}_1)$ in the proof were efficiently samplable* for each value of $\overline{r}_1$, then the *computational* indistinguishability of $\mathcal{X}'_{\lambda,\vec{x}}$ and $\mathcal{Y}'_{\lambda,\vec{x}}$ would imply $\mathcal{X}''_{\lambda,\vec{x}} \stackrel{c}{\equiv} \mathcal{Y}''_{\lambda,\vec{x}}$, which means the desired conclusion that the protocol after the use of the PRG is still secure. However, in fact $s(\overline{r}_1)$ is *not* efficiently samplable in general, and our proof strategy fails at this point if the statistical security for the PRG is relaxed.

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. This work was supported by JST PRESTO Grant Number JPMJPR14E8, Japan.

References

[1] M. Agrawal, N. Kayal, N. Saxena: PRIMES is in P. Annals of Mathematics, vol.160, no.2, pp.781–793, 2004.

[2] G. Asharov, A. Jain, D. Wichs: Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. IACR Cryptology ePrint Archive 2011/613 (`http://eprint.iacr.org/2011/613`), 2011 (Version 20120611:173056).

[3] 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).

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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).

[12] 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).

[13] 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.

[14] 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).

[15] M. Matsumoto, T. Nishimura: Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator. ACM Transactions on Modeling and Computer Simulation, vol.8, no.1, pp.3–30, 1998.

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

[17] 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.

Appendix

A Proof of Proposition 1

Here we give a proof of Proposition 1 in Section 3.1. Recall that the statement to be proved is: If a PKE scheme is CPA secure against non-uniform adversaries, then for any PPT and computationally secure PRG $\mathcal{R} = \mathcal{R}(1^\lambda, \mathsf{pk}, m)$ parameterized by a public key pk and a plaintext m, the PKE scheme is also CPA secure under the PRG $\mathcal{R}$ against non-uniform adversaries.

By the assumption on $\mathcal{R}$, the family of probability distributions $\left(\mathcal{R}(1^\lambda, \overline{\mathsf{pk}}, \overline{m})\right)_{\lambda,(\overline{\mathsf{pk}},\overline{m})}$ is computationally indistinguishable from the uniform distribution. Let $\mathcal{A}$ be any PPT non-uniform adversary with advice $z = z_\lambda$. By using the $\mathcal{A}$, we define two distinguishers $\mathcal{D}_0$ and $\mathcal{D}_1$ for the PRG $\mathcal{R}$ with various advice of the form $\hat{z} = \hat{z}_\lambda = (z_\lambda, \hat{\mathsf{pk}}_\lambda, \hat{m}_\lambda, \hat{\mathsf{st}}_\lambda)$ as follows, where $\hat{b} \in \{0,1\}$:

- Given an input R that is either uniformly random or a random output of $\mathcal{R}(1^\lambda, \overline{\mathsf{pk}}, \overline{m})$, $\mathcal{D}_{\hat{b}}$ generates $c^* \leftarrow \mathsf{Enc}(\hat{\mathsf{pk}}_\lambda, \hat{m}_\lambda; R)$ and $b \leftarrow \mathcal{A}^{(z_\lambda)}(\mathsf{guess}, 1^\lambda, \hat{\mathsf{st}}_\lambda, \hat{\mathsf{pk}}_\lambda, c^*)$, and then outputs 1 if $b = \hat{b}$ and 0 if $b \neq \hat{b}$.

Both $\mathcal{D}_0$ and $\mathcal{D}_1$ are PPT as Enc and $\mathcal{A}$ are PPT. In particular, the number of the possible advice $\hat{z}_\lambda$ is essentially finite as only a bounded part of $\hat{z}_\lambda$ is read by these algorithms. Therefore, for each λ and each $\hat{b} \in \{0,1\}$, there exists advice $z^\dagger_{\hat{b},\lambda} = (z_\lambda, \mathsf{pk}^\dagger_{\hat{b},\lambda}, m^\dagger_{\hat{b},\lambda}, \mathsf{st}^\dagger_{\hat{b},\lambda})$ that maximizes the upper bound at the given security parameter λ for the statistical distance between the output distributions of $\mathcal{D}_{\hat{b}}^{(\hat{z}_\lambda)}(1^\lambda, R)$ for the two choices of R. Since $\mathcal{D}_0$ and $\mathcal{D}_1$ are PPT and $\mathcal{R}$ is computationally secure, there exists a negligible function $\varepsilon(\lambda)$ satisfying that the statistical distance between the output distributions of $\mathcal{D}_{\hat{b}}^{(z^\dagger_{\hat{b},\lambda})}(1^\lambda, R)$ for the two choices of R is at most $\varepsilon(\lambda)$ for any λ and any $\hat{b}$. By the choice of the advice $z^\dagger_{\hat{b},\lambda}$, it then follows that the statistical distance between the output distributions of $\mathcal{D}_{\hat{b}}^{(\hat{z}_\lambda)}(1^\lambda, R)$ for the two choices of R is also bounded by $\varepsilon(\lambda)$ for any $\hat{b}$ and any advice $\hat{z}_\lambda$ of the form above.

Now we have

$$\begin{aligned}\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda) &= \mathrm{Ex}_{\mathsf{pk},m_0,m_1,\mathsf{st}}\left[\Pr\left[b = b^* : \begin{array}{c} b^* \xleftarrow{u} \{0,1\};\ c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{b^*}; \mathcal{R}(1^\lambda, \mathsf{pk}, m_{b^*}));\\ b \leftarrow \mathcal{A}(\mathsf{guess}, 1^\lambda, \mathsf{st}, \mathsf{pk}, c^*)\end{array}\right]\right] \\ &= \mathrm{Ex}_{\mathsf{pk},m_0,m_1,\mathsf{st}}\left[\frac{1}{2}\sum_{b^*\in\{0,1\}} \Pr\left[b = b^* : \begin{array}{c} c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{b^*}; \mathcal{R}(1^\lambda, \mathsf{pk}, m_{b^*}));\\ b \leftarrow \mathcal{A}(\mathsf{guess}, 1^\lambda, \mathsf{st}, \mathsf{pk}, c^*)\end{array}\right]\right]\end{aligned}$$

where the expectation value is taken over the probabilistic choice of $\mathsf{pk}, m_0, m_1, \mathsf{st}$ with $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{submit}, 1^\lambda, \mathsf{pk})$. By the definition of the distinguishers above, the quantity above can be expressed as

$$\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda) = \mathrm{Ex}_{\mathsf{pk},m_0,m_1,\mathsf{st}}\left[\frac{1}{2}\sum_{b^*\in\{0,1\}} \Pr\left[D_{b^*}^{(z_\lambda,\mathsf{pk},m_{b^*},\mathsf{st})}(1^\lambda, \mathcal{R}(1^\lambda, \mathsf{pk}, m_{b^*})) = 1\right]\right] .$$

Then by the triangle inequality, we have

$$\left|\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda)-\frac{1}{2}\right| \le \mathrm{Ex}_{\mathsf{pk},m_0,m_1,\mathsf{st}}\left[\frac{1}{2}\sum_{b^*\in\{0,1\}}\left|\begin{array}{c}\Pr\left[D^{(z_\lambda,\mathsf{pk},m_{b^*},\mathsf{st})}_{b^*}(1^\lambda,\mathcal{R}(1^\lambda,\mathsf{pk},m_{b^*}))=1\right]\\ -\Pr\left[D^{(z_\lambda,\mathsf{pk},m_{b^*},\mathsf{st})}_{b^*}(1^\lambda,U)=1\right]\end{array}\right|\right]$$
$$+\left|\mathrm{Ex}_{\mathsf{pk},m_0,m_1,\mathsf{st}}\left[\frac{1}{2}\sum_{b^*\in\{0,1\}}\Pr\left[D^{(z_\lambda,\mathsf{pk},m_{b^*},\mathsf{st})}_{b^*}(1^\lambda,U)=1\right]\right]-\frac{1}{2}\right|$$

where U denotes the uniform distribution. Now for the first term in the right-hand side, we have

$$\left|\Pr\left[D^{(z_\lambda,\mathsf{pk},m_{b^*},\mathsf{st})}_{b^*}(1^\lambda,\mathcal{R}(1^\lambda,\mathsf{pk},m_{b^*}))=1\right]-\Pr\left[D^{(z_\lambda,\mathsf{pk},m_{b^*},\mathsf{st})}_{b^*}(1^\lambda,U)=1\right]\right|\le\varepsilon(\lambda)$$

by the aforementioned choice of $\varepsilon(\lambda)$. This implies that the first term above is bounded by

$$\mathrm{Ex}_{\mathsf{pk},m_0,m_1,\mathsf{st}}\left[\frac{1}{2}\sum_{b^*\in\{0,1\}}\varepsilon(\lambda)\right]=\varepsilon(\lambda)\ .$$

On the other hand, for the second term above, we have

$$\mathrm{Ex}_{\mathsf{pk},m_0,m_1,\mathsf{st}}\left[\frac{1}{2}\sum_{b^*\in\{0,1\}}\Pr\left[D^{(z_\lambda,\mathsf{pk},m_{b^*},\mathsf{st})}_{b^*}(1^\lambda,U)=1\right]\right]=\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A}}(\lambda)$$

by the same argument as above. Now it follows that

$$\left|\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda)-\frac{1}{2}\right|\le\varepsilon(\lambda)+\left|\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A}}(\lambda)-\frac{1}{2}\right|\ ,$$

while $\left|\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A}}(\lambda)-1/2\right|$ is negligible in λ by the assumption on the CPA security of the PKE scheme (under the ideal internal randomness). This implies that $\left|\mathsf{Succ}^{\mathrm{CPA}}_{\mathcal{A};\mathcal{R}}(\lambda)-1/2\right|$ is also negligible in λ. This completes the proof.

B Another Counterexample for the Robustness

In this section, we give a pair of a two-party protocol π (see Section B.2) and a PRG $\mathcal{R}$ for Party $\mathcal{P}_1$ (see Section B.3) that shows that the "raw random tape" assumption on the simulator cannot be removed from the statement of Theorem 2. Namely, the protocol π securely computes a certain functionality f against semi-honest $\mathcal{P}_1$ with statistically indistinguishable simulator, the PRG $\mathcal{R}$ is statistically secure, but the simulator for $\mathcal{P}_1$ is not with raw random tape and the protocol $\pi\circ_1\mathcal{R}$ is not secure against semi-honest $\mathcal{P}_1$. We note that the protocol π below is also secure against the other party $\mathcal{P}_2$ in the semi-honest model.

B.1 Preliminaries: The Rabin Function

Here we summarize some facts about the Rabin function [16] 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 [16] that factoring the composite N is polynomial-time reducible to inverting Rabin function modulo the N and vice versa. Let $\mathsf{QR}_N \stackrel{\mathrm{def}}{=} \{x^2 \bmod N \mid x\in(\mathbb{Z}/N\mathbb{Z})^\times\}$, 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. Here we prepare a lemma:

Lemma 3. *Let p, q be two distinct λ-bit primes (i.e., $p, q \in \{2^{\lambda-1}, \ldots, 2^{\lambda}-1\}$), 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

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

Hence the assertion holds. □

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. Since $p \equiv q \equiv 3 \pmod 4$, 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 $x_p \xleftarrow{u} \{z_p, -z_p\}$ and $x_q \xleftarrow{u} \{z_q, -z_q\}$. (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.

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 as 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 in the Rabin function and let λ be the common bit length of p and q. We consider the following algorithm, which is given 1^{λ} and the N as inputs but *not* given any of the 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 $r \xleftarrow{u} \{0,1\}^{3\lambda}$, 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 $r \xleftarrow{u} \{0,1\}^{3\lambda}$, 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.

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 3 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 $a' \xleftarrow{u} (\mathbb{Z}/N\mathbb{Z})^\times$, 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 3 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 these bounds for $Q_{x,y}/P_y$ from the probability $1/4$ of the uniformly random choice are evaluated as

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

and

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

(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 $y \leftarrow r_1 \bmod N$ and sends N and y to $\mathcal{P}_2$.
2. $\mathcal{P}_2$ decides if $y \in \mathsf{QR}_N \subset (\mathbb{Z}/N\mathbb{Z})^\times$ 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 B.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 2: Another two-party protocol for our counterexample

where we used the relations $(1-\rho_1')2^{-3\lambda} \leq \delta \leq 2^{-3\lambda}$ and

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

Both of those two upper bounds for $|Q_{x,y}/P_y - 1/4|$ are negligible in λ and are 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, $\left(\left(\frac{z}{p}\right), \left(\frac{z}{q}\right)\right) = (1,1)$ for $z = 1$; $(1,-1)$ for $z = a$; $(-1,1)$ for $z = -a$; $(-1,-1)$ 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. □

B.2 The Protocol

Here we construct the aforementioned two-party protocol π as in Figure 2.[6] The security against $\mathcal{P}_2$ is now trivial since $\mathcal{P}_2$ knows the secret input $N = pq$ for $\mathcal{P}_1$. On the other hand, we have the following:

Proposition 4. *The protocol π is secure against semi-honest $\mathcal{P}_1$ with statistically indistinguishable 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 2 implies that the output distribution of a probabilistic function $g(y) = y + (u \bmod K_y)\cdot N$ with $u \xleftarrow{u} \{0,1\}^{4\lambda}$, where $K_y = \lfloor (2^{3\lambda}-1-y)/N \rfloor + 1$, is statistically indistinguishable from the conditional distribution of 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 $U(\mathbb{Z}/N\mathbb{Z}) \stackrel{s}{\equiv} U((\mathbb{Z}/N\mathbb{Z})^\times)$ (with bound dependent solely on λ) owing to Lemmas 1 and 3. 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

[6] One may feel that this protocol is too unrealistic since, not just it outputs nothing, but it seems to be infeasible for each party to even verify the correctness of the local input. In fact, by using the AKS deterministic primality test [1] (for checking if p and q are primes) and an appropriate secure equality test protocol (for checking if $N = pq$), we can modify the protocol in a way that the local inputs for $\mathcal{P}_1$ and $\mathcal{P}_2$ are any integer N and any pair of integers (p,q) of appropriate lengths, respectively, and the protocol behaves in the same way as the original protocol if the conditions for inputs of the original protocol are satisfied, or else the protocol aborts.

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 $(x', y') \stackrel{s}{\equiv} (x, y)$, therefore $(r_1, x, y, \eta) \stackrel{s}{\equiv} (g(y), x, y, \eta) \stackrel{s}{\equiv} (g(y'), x', y', \eta')$. 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 indistinguishable (i.e., $(g(y'), \eta') \stackrel{s}{\equiv} (r_1, \eta)$) by the argument above. This completes the proof. □

B.3 The PRG

Here we construct the aforementioned statistically secure PRG $\mathcal{R}$ for Party $\mathcal{P}_1$ in the two-party protocol π in Section B.2 with the property that the protocol $\pi \circ_1 \mathcal{R}$ is *not secure against semi-honest* $\mathcal{P}_1$ (though π itself is secure as shown in Proposition 4 above). The construction of $\mathcal{R}$ is as follows:

- Given security parameter 1^λ and an input N for Party $\mathcal{P}_1$ in π as well as a random seed $\widetilde{r}$, $\mathcal{R}$ runs $\mathcal{S}_1(1^\lambda, N; \widetilde{r})$ where $\mathcal{S}_1$ is the simulator for $\mathcal{P}_1$ constructed in Proposition 4, obtains its output $(N, r_1, \mathsf{trans}_1)$, and then outputs r_1.

Since the simulator $\mathcal{S}_1$ for $\mathcal{P}_1$ is statistically indistinguishable by Proposition 4, the output of $\mathcal{R}$ is also statistically indistinguishable from the original random tape for $\mathcal{P}_1$, hence $\mathcal{R}$ is statistically secure.

From now, we show that the protocol $\pi \circ_1 \mathcal{R}$ is indeed not secure against semi-honest $\mathcal{P}_1$. The next result means that a semi-honest $\mathcal{P}_1$ can infer some non-trivial secret information on the input for $\mathcal{P}_2$ in the protocol $\pi \circ_1 \mathcal{R}$, which was certainly concealed by the original protocol π:

Proposition 5. *For any input pair $(N, (p, q))$ for the two parties $\mathcal{P}_1, \mathcal{P}_2$, Party $\mathcal{P}_1$ can efficiently determine the secret input (p, q) for Party $\mathcal{P}_2$ with probability at least $1/16 - \mathsf{negl}(\lambda)$ by using the view for $\mathcal{P}_1$ in the protocol $\pi \circ_1 \mathcal{R}$, where negl denotes some negligible function.*

Proof. We note that, the elements output by $\mathcal{S}_1$ executed internally in $\mathcal{R}$ during an execution of $\pi \circ_1 \mathcal{R}$ can be deterministically recovered from the random tape $\widetilde{r}_1$ for $\mathcal{P}_1$ in $\pi \circ_1 \mathcal{R}$. Now the output of $\mathcal{S}_1$, including a simulated random tape r_1' and a simulated transcript η (either an element $x' \in (\mathbb{Z}/N\mathbb{Z})^\times$ or $\perp$) for $\mathcal{P}_1$ in the original protocol π, is statistically indistinguishable from the view for $\mathcal{P}_1$ in an execution of π. On the other hand, in an execution of π, the element y computed (deterministically) from the random tape for $\mathcal{P}_1$ at Step 1 is an element of QR_N with probability at least $1/4 - \mathsf{negl}'(\lambda)$ for some negligible function negl' and, provided $y \in \mathsf{QR}_N$, the transcript for $\mathcal{P}_1$ is an element x with $x^2 = y$. This implies that, for the corresponding element y' computed from r_1' in $\pi \circ_1 \mathcal{R}$, the probability (taken over the choice of $\widetilde{r}_1$) that $y' \in \mathsf{QR}_N$, $\eta = x' \in (\mathbb{Z}/N\mathbb{Z})^\times$ and $x'^2 = y'$ is also at least $1/4 - \mathsf{negl}''(\lambda)$ for some negligible function negl''.

For any $\widetilde{r}_1$ that yields $y' \in \mathsf{QR}_N$ and $x'^2 = y'$, Party $\mathcal{P}_2$ receives the y' in Step 1 of π internally executed by $\pi \circ_1 \mathcal{R}$, and then $\mathcal{P}_2$ sends to $\mathcal{P}_1$ one of the four square roots of y' chosen uniformly at random, denoted here by x''. In particular, the choice of x'' conditioned on the y' is independent of x'. Therefore, we have $x'' \notin \{x', -x'\}$ with probability $1/2$; and for any such x'', we have $x'^2 - x''^2 = (x' + x'')(x' - x'') = 0$ in $\mathbb{Z}/N\mathbb{Z}$ and $x' \pm x'' \neq 0$ in $\mathbb{Z}/N\mathbb{Z}$, which implies that $\gcd(x' - x'', N)$ is one of the two prime factors of N. Hence, given such an x'' (as well as x'), $\mathcal{P}_1$ can efficiently extract the *unordered set* $\{p, q\}$ of two prime factors of N, and then $\mathcal{P}_1$ can correctly guess the *order* of the pair (p, q) of the input for $\mathcal{P}_2$ with probability $1/2$. Summarizing, the probability that $\mathcal{P}_1$ determines the input for $\mathcal{P}_2$ correctly is at least $(1/4 - \mathsf{negl}''(\lambda)) \cdot 1/2 \cdot 1/2 = 1/16 - \mathsf{negl}''(\lambda)/4$. This completes the proof. □

Given only the local input N, it is computationally hard (unless factoring the N is easy) for $\mathcal{P}_1$ to specify the input for $\mathcal{P}_2$ which is the pair of the prime factors of N. On the other hand, by Proposition 5, the message sent from $\mathcal{P}_2$ during the protocol $\pi \circ_1 \mathcal{R}$ enables $\mathcal{P}_1$ to determine the input for $\mathcal{P}_2$ with almost constant probability. Therefore, the protocol $\pi \circ_1 \mathcal{R}$ should never be regarded as secure against the $\mathcal{P}_1$ (unless the integer factoring is easy). In fact, we have the following result:

Proposition 6. *If the protocol $\pi \circ_1 \mathcal{R}$ above is secure against semi-honest $\mathcal{P}_1$, then there exists a PPT algorithm that factorizes any integer $N = pq$ as in the Rabin function with probability at least $1/8 - \mathsf{negl}(\lambda)$ where negl denotes some negligible function.*

Proof. Let $\widetilde{\mathcal{S}}_1$ be the PPT simulator for Party $\mathcal{P}_1$ yielded by the assumption on the security of $\pi \circ_1 \mathcal{R}$. We consider the following algorithm $\mathcal{A}$: Given 1^λ and N as inputs, $\mathcal{A}$ first runs $\widetilde{\mathcal{S}}_1(1^\lambda, N)$ and obtains $(N, \overline{r}_1, \overline{x})$ where either $\overline{x} \in (\mathbb{Z}/N\mathbb{Z})^\times$ or $\overline{x} = \perp$. Secondly, $\mathcal{A}$ runs $\mathcal{S}_1(1^\lambda, N; \overline{r}_1)$ and obtains (N, r_1'', x'') where either $x'' \in (\mathbb{Z}/N\mathbb{Z})^\times$ or $x'' = \perp$. Now $\mathcal{A}$ aborts unless $\overline{x} \in (\mathbb{Z}/N\mathbb{Z})^\times$, $x'' \in (\mathbb{Z}/N\mathbb{Z})^\times$ and $x'' \notin \{\overline{x}, -\overline{x}\}$. Finally, $\mathcal{A}$ computes $P = \gcd(\overline{x} - x'', N)$ and outputs P and N/P. Note that $\mathcal{A}$ is a PPT algorithm.

Now we consider the following distinguisher $\mathcal{D}$ for simulator $\widetilde{\mathcal{S}}_1$: Given $(1^\lambda, N, \widehat{r}_1, \widehat{x})$ as input, $\mathcal{D}$ runs $\mathcal{S}_1(1^\lambda, N; \widehat{r}_1)$ and obtains $(N, \widehat{r}_1', \widehat{x}')$; $\mathcal{D}$ computes $\widehat{y} \in (\mathbb{Z}/N\mathbb{Z})^\times$ as in Step 1 of π by using $\widehat{r}_1'$ as the random tape for $\mathcal{P}_1$; and then $\mathcal{D}$ outputs 1 if $\widehat{x}, \widehat{x}' \in (\mathbb{Z}/N\mathbb{Z})^\times$, $\widehat{x}^2 = \widehat{x}'^2 = \widehat{y}$ and $\widehat{x} \notin \{\widehat{x}', -\widehat{x}'\}$, or else outputs 0. Note that $\mathcal{D}$ is a deterministic polynomial-time algorithm. Since the output of $\widetilde{\mathcal{S}}_1$ is computationally indistinguishable from the view of $\mathcal{P}_1$ in an execution of $\pi \circ_1 \mathcal{R}$, $\Pr\left[\mathcal{D}(1^\lambda, N, \overline{r}_1, \overline{x}) = 1\right]$ has only negligible difference from $\Pr\left[\mathcal{D}(1^\lambda, N, \widetilde{r}_1, x) = 1\right]$, where $\widetilde{r}_1$ is a uniformly random tape for $\mathcal{P}_1$ in $\pi \circ_1 \mathcal{R}$ and x is the transcript for $\mathcal{P}_1$ in an execution of $\pi \circ_1 \mathcal{R}$ corresponding to the random tape $\widetilde{r}_1$.

For an execution of $\pi \circ_1 \mathcal{R}$, write $\mathcal{S}_1(1^\lambda, N; \widetilde{r}_1) = (N, r_1', x')$, and let y' be the element of $(\mathbb{Z}/N\mathbb{Z})^\times$ computed as in Step 1 of π by using r_1' as the random tape for $\mathcal{P}_1$. Then, since the simulator $\mathcal{S}_1$ for $\mathcal{P}_1$ in π is statistically indistinguishable, it follows by the property of π that we have $y' \in \mathsf{QR}_N$ and $x'^2 = y'$ with probability at least $1/4 - \mathsf{negl}'(\lambda)$ for some negligible function negl'. Moreover, conditioned on these y' and x', the construction of $\pi \circ_1 \mathcal{R}$ implies that, for the corresponding elements y and x appeared in an execution of $\pi \circ_1 \mathcal{R}$ with the $\widetilde{r}_1$ as the random tape for $\mathcal{P}_1$, we always have $y = y'$ since y is dependent solely on the same random tape $\widetilde{r}_1$ for $\mathcal{P}_1$, and x is uniformly random over the four square roots of $y = y'$, which differs from $\pm x'$ with probability $1/2$. Hence, $\mathcal{D}(1^\lambda, N, \widetilde{r}_1, x)$ outputs 1 with probability at least $(1/4 - \mathsf{negl}'(\lambda)) \cdot 1/2 = 1/8 - \mathsf{negl}'(\lambda)/2$.

This implies that $\Pr\left[\mathcal{D}(1^\lambda, N, \overline{r}_1, \overline{x}) = 1\right] \geq 1/8 - \mathsf{negl}''(\lambda)$ for some negligible function negl''. Moreover, by the construction of $\mathcal{D}$, the fact $\mathcal{D}(1^\lambda, N, \overline{r}_1, \overline{x}) = 1$ guarantees that $\mathcal{A}(1^\lambda, N)$ does not abort and correctly outputs the two factors of N (similarly to the proof of Proposition 5). This completes the proof. □