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

November 4, 2016

Abstract

It is widely understood that we are just human beings rather than being almighty; as a result, when utilizing cryptographic technology in practice, we have always to rely on imperfect randomness to various extent. For ordinary cryptography (e.g., encryption), if a person misuses a random number generator (e.g., generating a vulnerable ciphertext), then the person him/herself will suffer the damage caused by the misuse (e.g., leakage of his/her sensitive plaintext). In contrast, in this paper we give *a concrete example* of the following serious situation for multiparty (in particular, two-party) computation in the semi-honest model: Imagine that a party replaces the party's (uniform and independent) internal random tape with an output of some random function. Then, even if the original two-party protocol is *statistically* (i.e., almost perfectly) secure and the output of the random function is *statistically* (i.e., almost perfectly) indistinguishable from uniform, *it may still happen* that the party with the replaced random tape can now get *the other party's secret input*. Due to the unavoidable imperfect randomness mentioned above, statistical indistinguishability would be the "best possible" quality of the random tape in practice, but then, our example may mean that the semi-honest security alone of a two-party protocol can guarantee nothing about security for a party's input in real situations.

A technical remark is that, the output of the random function in our example depends also on the party's input for our two-party protocol. But the author thinks that the true reason of such a paradoxical phenomenon is at another point. Namely, there is a famous "clever" technique for security proofs of two-party protocols, where a simulator for a party's view emulates a transcript during the protocol first, and then emulates the party's random tape depending consistently on the transcript. This is actually in the opposite order from real, where the transcript does depend on the random tape; and, in fact, we prove that such a problem caused by "manipulated" random tapes is prevented (at least under the two conditions for statistical security mentioned above) *provided the simulator in the original security proof is constructed without this "order-reversing" technique*, even if the output of the random function still depends on the party's local input.

1 Introduction

The difference between the ideal random numbers in theoretical models and the non-ideal (or sometimes even terrible) random numbers in the real world is always a major worrying problem in the theory and applications of cryptographic technology. 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 that were caused by inappropriate generation of random cryptographic keys. This is in fact a kind of "worst" example, but in general, it seems to be even beyond the art

of human being to generate an ideal random number (moreover, on an accurate, "non-random" digital computer). 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 [16]).

There are mainly two directions of efforts towards achieving security in the real world by using non-ideal random numbers available for human. One direction, from the viewpoint of cryptographic researchers, is to theoretically design a cryptographic scheme and prove its security, under a hypothesis on the use of a kind of *imperfect* randomness. Here we refer to only a few papers [5, 6] from the very wide research area of possibility and impossibility of security with imperfect randomness. The other direction is to try to realize random numbers as close to ideal as possible, by either (or combining both of) software-oriented strategies (i.e., PRGs) or utilizing entropy originated from the aspect of real computers being hardware devices (e.g., thermal noise arising during the run) or from outside of computers (e.g., timing of program execution or rhythm of users' keystrokes).

In any case, usually a designer of a cryptographic scheme puts some theoretical hypothesis on the quality of random numbers (e.g., to be ideally random) used by an implementation of the scheme, and then we (or, at least some theoretical cryptographers) would think that a user of the system may still utilize "bad" random numbers not following the scheme's specification, but *only at the user's own risk*. This paradigm would be (at least to some extent) reasonable for most cases of "single-party" cryptographic primitives. For example, in an encryption scheme, if a user encrypts his/her sensitive information by using low-quality random numbers, which will enable an adversary to reveal the encrypted information, then the person who suffers the damage of the information leakage is the user him/herself who misused the random numbers. This can be a motivation for a user to utilize as "good" random numbers as possible when encrypting his/her data.

Now we move to the case of *secure multiparty computation*; to simplify the argument, we focus on *two-party computation* in this paper, where each of the two parties has his/her local input and they want to compute the value of some function of their inputs by communicating with each other *while keeping each party's local input still secret against the other party* (except any information implied by the function value itself). Several security models (the semi-honest model, the malicious model, etc.) for two-party computation have been proposed depending on the supposed level of trust in each party, and it is at least theoretically feasible (with a certain cryptographic assumption, if any) to perform secure two-party computation for any function under each security model (see e.g., Chapter 7 of Goldreich's book [7], in particular Theorem 7.4.1 in Section 7.4). Moreover, by virtue of its high practical usefulness, the topic of secure two-party computation has recently been attracting interests from broader communities not only within cryptography or mathematics; e.g., we refer to two recent papers [19, 20] from major journals in the area of bioinformatics.

Anyway, even in a two-party computation protocol, if a party utilizes "bad" random numbers during the joint computation and the misuse of random numbers causes leakage of *this party's* secret input to the other party, then this would still be within the aforementioned "at the user's own risk" paradigm. But how about the opposite case, where a party's misuse of random numbers causes leakage of *the other party's* secret input (*to the party of originally misusing the random numbers*)? This problem is the subject of the present paper.

1.1 Our Results

Briefly summarizing, our main result in this paper is to give a *concrete example* of a pair of a (polynomial-time executable) two-party computation protocol π (between two parties $\mathcal{P}_1$ and $\mathcal{P}_2$) and a (polynomial-time computable) function $\mathcal{F}$ with the following "paradoxical" property:

- The protocol π *is secure against semi-honest* $\mathcal{P}_1$ in the standard sense (cf. Section 3.1). We note that π is also secure against semi-honest $\mathcal{P}_2$ (under some computational assumption).
- The output distribution of the function $\mathcal{F}$ with uniformly random input is *statistically indistinguishable* from the internal (uniformly) random tape for $\mathcal{P}_1$.
- However, the protocol π *becomes NOT secure against* $\mathcal{P}_1$ even when $\mathcal{P}_1$ only utilizes the random output of $\mathcal{F}$ as the internal random tape for π, which is even *statistically indistinguishable from the original random tape*, and thereafter behaves as a semi-honest party. More precisely, in this case, $\mathcal{P}_1$ can know the secret input for the other party $\mathcal{P}_2$ with high probability. (See Section 3.2 for our formulation of "security against a party with randomness preparation".)

See Section 4.1 and Section 4.2 for the construction of the π and $\mathcal{F}$, respectively. The other major result in this paper will be explained in Section 1.2 below.

The *semi-honest model* is one of the most standard theoretical models for secure two-party computation, where each party is supposed to behave during a joint execution of a protocol exactly as specified by the protocol (without e.g., sending some maliciously forged message to the other party at a step), while a semi-honest party may try to infer some non-trivial information on the other party's input (that is not directly implied from the local input and the local output for the party) from all the data viewed by the party during the protocol execution. Then the security against a semi-honest party is defined to mean that, the "view" by the party during the protocol execution can be (polynomial-time) simulated in a (computationally, or sometimes even statistically) indistinguishable manner, from the input and output for the party only; this then guarantees that, any information that can be inferred during a real protocol execution can also be inferred from the party's input and output only, which is thus only trivial information. Here we clarify that, the "view" by the party formally consists of the party's local input, *the content of the party's internal random tape* used in the protocol execution, and all the messages sent from the other party during the protocol. The most important point here is that *a party's view involves the party's random tape*, to be mentioned below several times again. The security against semi-honest party is widely regarded as a "minimal security requirement" for a two-party computation protocol to be practically reasonable. (We note that the semi-honest security is in fact not enough in many practical applications of two-party computation, and hence other security models, such as the malicious model, have also been proposed and widely studied; see below for further comments.)

We note that the semi-honest model originally supposes each party to use a perfectly uniform and independent random tape, therefore our result mentioned above is not a contradiction *in theory*; the "attack" by a party utilizing an output of a function $\mathcal{F}$ instead of the original random tape is "out of scope" of the semi-honest security of the original protocol π. However, it is practically infeasible as mentioned above to generate an ideally random tape, therefore using "statistically indistinguishable from ideal" random tapes would be a "best possible" way *in practice*. From this standpoint, whenever we allow the use of such "best possible" random tapes, our example would mean that *the semi-honest security alone of a protocol can guarantee nothing about the security in a practical situation*, and it cannot be excused by the aforementioned "by the user's own risk" paradigm. (It should be remarked here that, the random output of our function $\mathcal{F}$ in fact depends also on the party's local input for the protocol π; see Section 1.2 for further discussion on this.)

1.2 Discussion on Impact of Our "Counterexample" Protocol

Here we discuss how our results in this paper, especially the "counterexample" mentioned in Section 1.1, are meaningful or are worth considering as a real problem for secure two-party computation.

First of all, we mention that our "counterexample" does *not* originate in any technical intricacy to formalize computational security of two-party computation, in particular, not originate in the use of non-uniform distinguishers to define the computational indistinguishability of simulator's output from the real view. Indeed, the simulator for the view of $\mathcal{P}_1$ in the original protocol π is in fact *statistically* indistinguishable (see Theorem 1), while the output distribution of $\mathcal{F}$ is also *statistically* indistinguishable as mentioned above. (We note that, we actually relies on the computational hardness of integer factoring to show that, the information on the input for $\mathcal{P}_2$ that can be inferred by using the function $\mathcal{F}$ is indeed *non-trivial* information; cf. Theorem 4.)

One may think of the fact (mentioned above), that the output of the function $\mathcal{F}$ to prepare the new random tape for $\mathcal{P}_1$ depends also on the input for $\mathcal{P}_1$, as the source of our problem here. But the author of the paper thinks that this point would not be the main origin of the current problem, by some reasons explained later. Instead, here we revisit the following famous technique in security proofs for two-party computation. That is, one may construct a simulator for a party's view by emulating the transcript part (i.e., messages sent from the other party) first and then adjusting the random tape part to be consistent with the transcript, opposite to the "true order"; a party in a real protocol execution generates the random tape first, and then executes the protocol (and receives the messages from the other party) according to the chosen random tape. Such a construction of simulators seems to be regarded by experts as a "clever solution" rather than a kind of "bad know-how". Nevertheless, this proof technique may cause a problem when considering random tapes generated by some function, say the $\mathcal{F}$ above; that is, in this modified situation, such a simulator has to adjust the random tape part to *both* the transcript *and the value of* $\mathcal{F}$ *with randomly chosen input simultaneously*. In fact, the construction of simulator for $\mathcal{P}_1$ in our protocol π also relies on this technique (see the proof of Theorem 1); and we prove (as a corollary of a general Theorem 9 in Section 6) that the problem caused by utilizing the *statistically* close to uniformly random function $\mathcal{F}$ as in Section 1.1 would be prevented if the *statistically* indistinguishable simulator for $\mathcal{P}_1$ in the original protocol π were constructed without the "adjusting random tape" technique. (Here we emphasize that, this affirmative result holds even when the generated random tape depends on the input for $\mathcal{P}_1$ in the protocol as our function $\mathcal{F}$ indeed does.)

One may be curious about whether the two "statistically" conditions appeared at the end of the previous paragraph, in particular that for the random function $\mathcal{F}$, can be weakened to a seemingly reasonable "computationally indistinguishable" condition. An explanation of why we put the stronger conditions will be given in a discussion after the proof of Theorem 9. Here we give another explanation; imagine that the function $\mathcal{F}$ were a computationally secure PRG. Now, for example, in the CPA security game for public key encryption, a distinguisher can see the challenge ciphertext generated from a challenger's internal random tape, but *cannot* see the challenger's random tape itself; therefore, the challenger can replace his/her random tape with an output of such a PRG in an indistinguishable way. In contrast, in the case of two-party computation, a distinguisher for a simulator of a party's view is supposed to *be able to see* the party's random tape in a real protocol execution; when a party replaces the random tape with an output of a PRG, this ability of the distinguisher would also enable oneself to see *the seed of the PRG* as well (this observation is in fact reflected in our formulation in Section 3.2 of "security against randomness preparation function"), which would remove the power of computational security of the PRG.

There would be (at least) three more kinds of objections to the claim of this paper. The first one would be as follows: The dependence of the new random tape generated by the function $\mathcal{F}$ on the input for the party $\mathcal{P}_1$ still looks just an unrealistic, artificially introduced property. A possible answer is as follows: Consider the case where the operating system of a computer can generate (pseudo)random numbers from the rhythm of a user's keystrokes, and a party starts executing a two-party protocol by typing the input as an option to the program and then hitting the enter key.

In this case, even when the party is (semi-)honest, the party's rhythm to type the input string may depend on how the string is simple or complicated, which may result in the generated random tape for the protocol dependent on the input. This observation would explain that the input-dependent generation of the random tape may happen in practice, either maliciously or unintentionally.

The second objection would be as follows: The manipulation of internal random tapes looks an advanced (or close to fully malicious) attack, and it seems to be too strict to require a countermeasure against such attacks as "the minimal security requirement" like the semi-honest security at the present. To answer this, the author would like to refer to an example even from outside of cryptography — so-called "tool-assisted speedrun" (or "tool-assisted superplay", TAS), which is a kind of "joky" way of enjoying computer games. In TAS, a player does basically *not modify the game program itself* but, for example, just *watches the internal state for random number generation* in order to exhibit, by selecting a best timing of hitting a button, an excellent play that is still "theoretically possible on a real game machine". The following description for TAS is quoted from Wikipedia [21]: "... *also facilitates another common technique, luck manipulation, which is the practice of exploiting the game's use of player input in its pseudo-random number generation to make favorable outcomes happen.* ...". This example does or does not show that, manipulation of randomness by just watching the program's internal state may be mounted so casually, and it is not trivially reasonable to just ignore such a kind of attacks. Anyway, if the computer's random number generation depends on a user's previous behavior as in the example of the previous paragraph, then such an "attack" may happen in practice, either maliciously or even unintentionally.

The third (and the final) objection here would be as follows: The semi-honest model is originally too weak to capture the practically reasonable attacks, not only the one pointed out in this paper, and therefore we should have been moved to more realistic models rather than staying at adopting the semi-honest model, irrelevantly to this paper. This is in some sense reasonable, as we show in Section 5.4 that such a kind of attacks in this paper can be prevented if the original protocol π is secure in the *semi-malicious model*, which was introduced by Asharov, Jain, and Wichs in Section A.2 of [2] and is still weaker than the fully malicious model. Here we omit the detail of this model (cf. [2] or Section 5.4 of this paper), but this affirmative result is natural, as the semi-malicious model already captures such a kind of attacks by definition. (On the other hand, some other possible alternatives to the semi-honest model, namely the *augmented semi-honest model* introduced by Definition 7.4.24 in Section 7.4.4.1 of Goldreich's book [7] and its variant suggested in a short note by Hazay and Lindell [8], does not prevent this attack in general; see Section 5.2 and Section 5.3, respectively.) However, the semi-malicious model seems to be "overqualified" for resolving the problem in this paper. Namely, a semi-malicious party is (roughly speaking) allowed to also *modify the local input*, not only the random tape, even *adaptively during a protocol execution*; while in this paper, we are concerning modification of *random tape before a protocol execution only*. In contrast, our Theorem 9 mentioned above proposes sufficient conditions to prevent our attack, which (in particular, the condition of statistical indistinguishability) may look too strict but are still verifiable within the framework of the semi-honest model (instead of adopting a new model).

1.3 Related Work

There are some previous papers in which the same problem, that the security of a two-party protocol is not preserved in general when a party uses a PRG to generate the party's random tape in the protocol, has appeared, sometimes even implicitly; e.g., [14] (or Theorem 5.7 of [15]), [11] (or Section 4 of [12]), and Section 3.3 of [9]. In these papers, the authors seem to had aimed, in various settings, at proving a kind of lower bounds for communication complexity to realize some kinds of secure two-party computation. Their strategies seemed to be, first showing a lower bound

for "overall complexity" of such computation, and then cancelling the effect of the sizes of the parties' random tapes to the overall complexity by replacing the random tape with an output of a PRG (with significantly short random seed). However, the approach should fail at the latter step due to the problem mentioned above; accordingly, they have shown such lower bounds in a certain weaker form [9, 15] or under a different security model named "honest-but-deterministic" model [12]. (To be more precise, the authors of [12] considered the lack of such lower bounds affirmatively and developed a secure two-party protocol in the semi-honest model that achieves a certain kind of "surprising" functionality.) Among them, only the authors of [9] clearly mentioned where the problem is. However, even they did not gave a concrete example (as in the present paper) of that the use of even reasonably secure random generators compromises the security of a protocol.

There are also other recent studies on some problems caused by artificially manipulated PRGs, called "backdoored PRGs" (cf. [4]). This topic has some similarity to the present paper; they also concern a case where a party's manipulation of random number generators will cause leakage of another party's secret and this manipulation looks difficult to detect. However, the notion of backdoored PRGs focuses mainly on potential attacks to ordinary parties by a third party (such as NIST) who should be fully trustful; and hence the viewpoint is still different from our study.

2 Preliminaries

2.1 Basic Notations

In this paper, we write $\{0,1\}^* = \bigcup_{n\geq 0}\{0,1\}^n$. We say that a non-zero polynomial is *positive*, if any of its coefficients is non-negative. We say that a function $f(\lambda)$ of integers $\lambda \geq 1$ with $0 \leq f(\lambda) \leq 1$ is *negligible* in λ, if for any positive polynomial poly, there exists an integer $\lambda_0 \geq 1$ satisfying $f(\lambda) < \mathsf{poly}(\lambda)^{-1}$ for any $\lambda > \lambda_0$. We abuse a notation $\mathsf{negl} = \mathsf{negl}(\lambda)$ to mean some negligible value, which may vary depending on the situation. We write $a \leftarrow \mathcal{D}$ to signify that the element a is chosen according to a probability distribution $\mathcal{D}$. We define $\mathcal{U}(X)$ to be the uniform distribution on a set X, and let $a \leftarrow_R X$ mean $a \leftarrow \mathcal{U}(X)$.

In this paper, we abbreviate $a \equiv b \pmod n$ to $a \equiv_n b$. For any integers $M \leq N$, we define $[M,N] := \{a \in \mathbb{Z} \mid M \leq a \leq N\}$. We often identify an n-bit sequence with an integer in $[0, 2^n - 1]$ via the binary expression of integers, and sometimes identify an element $x \in \mathbb{Z}/N\mathbb{Z}$ with the integer $a \in [0, N-1]$ determined by $a \bmod N = x$. Let $(\mathbb{Z}/N\mathbb{Z})^\times$ denote the set of invertible elements in the ring $\mathbb{Z}/N\mathbb{Z}$, i.e., elements $a \bmod N$ where $a \in [1, N-1]$ is coprime to N. We use a notation such as $[F(r)\colon \mathcal{R}]$ to signify a random variable with value $F(r)$ where r follows the probability distribution specified by $\mathcal{R}$. For example, $[bb\colon b \leftarrow_R \{0,1\}]$ means $\mathcal{U}(\{00, 11\})$. We often abbreviate "probabilistic polynomial-time" to "PPT". For any probabilistic algorithm $\mathcal{A}$ with input x, we may write $\mathcal{A}(x; r)$ instead of $\mathcal{A}(x)$ in order to emphasize the choice of the random tape r used by $\mathcal{A}$. On the other hand, when $\mathcal{A}$ is a non-uniform algorithm with auxiliary advice z, we often make the z implicit in the notations, while we may write $\mathcal{A}^{(z)}$ to clarify the given advice z.

2.2 Indistinguishability of Random Variable Families

A standard definition of security for two-party computation relies on the notion of indistinguishability between two families of random variables against *non-uniform* distinguishers, which we recall as follows. Our formulation here is essentially the same as the standard one described, e.g., in Section 7.2.1.2 of [7] but the notations are slightly different. The setting here is: $(I_n)_{n\geq 1}$ is a family of subsets $I_n \subset \{0,1\}^*$ indexed by a positive integer n. $X = (X_{n,w})_{n,w}$ and $Y = (Y_{n,w})_{n,w}$ are

families of random variables $X_{n,w}$ and $Y_{n,w}$ indexed by a pair of $n \geq 1$ and $w \in I_n$. Then we describe the definition in the computationally bounded situation:

Definition 1 (Computational Indistinguishability)**.** Let $(I_n)_n$, $X = (X_{n,w})_{n,w}$ and $Y = (Y_{n,w})_{n,w}$ be as above. We say that X and Y are *computationally indistinguishable* and write $X \stackrel{c}{\equiv} Y$, if for any *non-uniform* PPT algorithm $\mathcal{D}$, there exists a negligible function $\mu(n)$ satisfying that, for any $n \geq 1$,

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

In order to describe the notion of statistical indistinguishability, first we recall the definition of the statistical distance between two probability distributions $\mathcal{X}, \mathcal{Y}$ over a common (finite) set Z. Their *statistical distance* $\Delta(\mathcal{X}, \mathcal{Y})$ is defined by

$$\Delta(\mathcal{X}, \mathcal{Y}) := \frac{1}{2} \sum_{z \in Z} \left| \Pr_{z' \leftarrow \mathcal{X}}[z' = z] - \Pr_{z' \leftarrow \mathcal{Y}}[z' = z] \right| \ .$$

It is known that $\Delta(\mathcal{X}, \mathcal{Y})$ is equal to the maximum of $\Pr_{z \leftarrow \mathcal{X}}[z \in A] - \Pr_{z \leftarrow \mathcal{Y}}[z \in A]$ taken over all the subsets $A \subset Z$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are *ε-close*, if $\Delta(\mathcal{X}, \mathcal{Y}) \leq \varepsilon$. It is important that we have $\Delta(f(\mathcal{X}), f(\mathcal{Y})) \leq \Delta(\mathcal{X}, \mathcal{Y})$ for any probabilistic function f independent of $\mathcal{X}$ and $\mathcal{Y}$. Now we describe the definition of the statistical indistinguishability, which is a "computationally unbounded variant" of Definition 1:

Definition 2 (Statistical Indistinguishability)**.** Let $(I_n)_n$, $X = (X_{n,w})_{n,w}$ and $Y = (Y_{n,w})_{n,w}$ be the same as in Definition 1. 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 $\mu(n)$ satisfying that, for any $n \geq 1$, $X_{n,w}$ and $Y_{n,w}$ are $\mu(n)$-close for every $w \in I_n$.

2.3 Approximately Uniform Sampling from Some Sets

Here we summarize some easy lemmas used below, stating a kind of facts that uniformly random sampling from a certain set can be approximated well by the output of some easy-to-compute function with efficiently samplable random input. In this subsection, let $a \bmod N$ mean the remainder of an integer a modulo a positive integer N, which takes a value in $[0, N-1]$.

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 $U_M := \mathcal{U}([0, M-1])$ and $U_N := \mathcal{U}([0, N-1])$. 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 $0 \leq a \leq M - \lfloor M/N \rfloor N - 1$ 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. □

Lemma 2. *Let L, M, N be positive integers with $L \leq N$. Let $a \in [0, L-1]$, and let $A := \{k \in [0, N-1] \mid k \bmod L = a\}$, $K := \lfloor (N-1-a)/L \rfloor + 1$, and $\delta := M/K - \lfloor M/K \rfloor$. Moreover, we set $U_M := \mathcal{U}([0, M-1])$. Then we have $a + (\ell \bmod K) \cdot L \in A$ for any $\ell \in [0, M-1]$, and*

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

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 $A = \{a + k \cdot L \mid k \in [0, K-1]\}$, which yields the first part of the assertion and also implies the second assertion by Lemma 1. □

Lemma 3. *Let p, q be two distinct primes of λ-bit length (i.e., $p, q \in [2^{\lambda-1}, 2^{\lambda}-1]$), and let $N := pq$. Moreover, we set $U := \mathcal{U}(\mathbb{Z}/N\mathbb{Z})$ and $U' := \mathcal{U}((\mathbb{Z}/N\mathbb{Z})^{\times})$. Then we have $\Delta(U, U') < 2^{-(\lambda-2)}$. Hence we also have $\Delta(f(U), U') < 2^{-(\lambda-2)}$ for any function f with domain $\mathbb{Z}/N\mathbb{Z}$ that is identical on the subset $(\mathbb{Z}/N\mathbb{Z})^{\times}$.*

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

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

Hence the assertion holds. □

2.4 The Paillier Cryptosystem

Here we summarize some facts about the Paillier cryptosystem [17] used in our argument below; we make slight modifications for the sake of convenience in later use while keeping the essence of [17]. The Paillier cryptosystem is a public key encryption scheme, which admits a PPT *key generation algorithm* Gen, a PPT *encryption algorithm* Enc, and a deterministic polynomial-time *decryption algorithm* Dec. Given a security parameter 1^{λ}, the algorithm $\mathsf{Gen}(1^{\lambda})$ first generates, in a certain suitable manner, two different primes p, q of the same $\rho(\lambda)$-bit length where $\rho = \rho(\lambda)$ is a certain positive polynomial with $\rho(\lambda) \geq \lambda$. Then the algorithm computes $N := pq$, and outputs a *public key* $\mathsf{pk} := N$ and a *secret key* $\mathsf{sk} := (p, q)$. Now the plaintext space is $\mathbb{Z}/N\mathbb{Z}$. Given 1^{λ}, pk and a plaintext $m \in \mathbb{Z}/N\mathbb{Z}$, the algorithm $\mathsf{Enc}_{\mathsf{pk}}(1^{\lambda}, m)$ first chooses a random $(4\rho(\lambda) + \lambda)$-bit sequence r_0 and computes $r := r_0 \bmod N^2$. If $r \notin (\mathbb{Z}/N^2\mathbb{Z})^{\times}$, then the algorithm replaces the r with 1. Secondly, the algorithm computes $(1+N)^m r^N \bmod N^2 \in (\mathbb{Z}/N^2\mathbb{Z})^{\times}$ and outputs it as a ciphertext of m, denoted often by $[[m]]$ in this paper. We note that, by Lemmas 1 and 3 and the fact that $N^2 \leq 2^{4\rho(\lambda)}$ and $\rho(\lambda) \geq \lambda$, if the bit sequence r_0 is uniformly random, then the distribution of r (respectively, of $[[m]]$) is statistically indistinguishable from the uniform distribution on $(\mathbb{Z}/N^2\mathbb{Z})^{\times}$ (respectively, on the set of ciphertexts of m) where a bound for the statistical distance is negligible in λ and dependent solely on λ. On the other hand, we omit the detail of the algorithm Dec (using sk) since it is not relevant to our argument, except the fact that Dec always outputs m for any given input ciphertext $[[m]]$ of m.

The Paillier cryptosystem admits a deterministic *additively homomorphic operation* that generates a ciphertext $[[m_1 + m_2]]$ from two ciphertexts $[[m_1]]$ and $[[m_2]]$; in fact, the operation is just the multiplication in $(\mathbb{Z}/N^2\mathbb{Z})^{\times}$ in this case. Moreover, given any ciphertext $[[m]]$, if a ciphertext $[[0]]$ is randomly generated as described above, then the result of the additively homomorphic operation

for the $[[m]]$ and $[[0]]$ is also statistically indistinguishable (with bound dependent solely on λ) from the uniform distribution on the set of ciphertexts of m that is independent of the original choice of $[[m]]$. We refer to this operation as the *statistically perfect re-randomization* of a ciphertext.

For the security, we require the indistinguishability against chosen plaintext attacks (CPA security) as usual, except that now the distinguisher may be a *non-uniform* PPT algorithm. To be precise, we consider the following security game between a challenger and a non-uniform PPT (with respect to λ) adversary:

1. The adversary is given the security parameter 1^λ and advice z associated to the parameter λ.
2. The challenger is also given the security parameter 1^λ, and then generates $(\mathsf{pk}, \mathsf{sk})$ by $\mathsf{Gen}(1^\lambda)$ and sends pk to the adversary.
3. The adversary chooses two plaintexts m_0, m_1 and sends these to the challenger.
4. Challenger chooses a uniformly random bit b, generates a challenge ciphertext $[[m_b]]$ by $\mathsf{Enc}_{\mathsf{pk}}(1^\lambda, m_b)$, and sends $[[m_b]]$ to the adversary.
5. Finally, the adversary generates a bit b'.

Then we assume that, for any non-uniform PPT adversary and any choice of advice, there exists a negligible function $\varepsilon = \varepsilon(\lambda)$ satisfying $|\Pr[b' = b] - 1/2| \leq \varepsilon$ in the game above for every λ.

2.5 The Rabin Function

Here we summarize some facts about the Rabin function [18] 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 $p \equiv_4 q \equiv_4 3$ (see Section 2.1 for notations). It is known [18] that factoring the composite N is polynomial-time reducible to inverting Rabin function modulo the N and vice versa. Let $\mathsf{QR}_N := \{x^2 \bmod N \mid x \in (\mathbb{Z}/N\mathbb{Z})^\times\}$, 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.

Now we recall the following fact for finding a square root 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 at random among the four square roots of y modulo N.*

Proof. Since $p \equiv_4 q \equiv_4 3$, both $p' := (p+1)/4$ and $q' := (q+1)/4$ are integers. The algorithm runs in the following four steps: (i) Compute $y_p := y \bmod p$, $y_q := y \bmod q$, $z_p := {y_p}^{p'}$, and $z_q := {y_q}^{q'}$. (ii) Choose $x_p \leftarrow_R \{z_p, -z_p\}$ and $x_q \leftarrow_R \{z_q, -z_q\}$. (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 above 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 $a := r \bmod N$ by using a uniformly random 3λ-bit sequence r, 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 $x := r \bmod N$ by using a uniformly random 3λ-bit sequence r, 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) owing to the 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 uniform over $(\mathbb{Z}/N\mathbb{Z})^\times$, where the bound of the statistical distance is dependent solely on λ.*
- *If $y \in \mathsf{QR}_N$, then the 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 the distribution of the element a and the uniform distribution on $(\mathbb{Z}/N\mathbb{Z})^\times$ 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 $a' \leftarrow_R (\mathbb{Z}/N\mathbb{Z})^\times$, 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 $\rho_1' := (1/2 + 2^{-(\lambda-1)})^\lambda$, 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 $2^{-(\lambda+2)} + 2^{-(\lambda-2)} < \rho_2' := 2^{-(\lambda-1)}$, the latter being negligible in λ and dependent solely on λ. Hence, considering throughout 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 $\alpha := (1 - \rho_1)\lfloor 2^{3\lambda}/N \rfloor / 2^{3\lambda}$ and $\delta := (1 - \rho_1)/2^{3\lambda}$. 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 $p \equiv_4 q \equiv_4 3$, 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 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

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

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 Lemma 5.

Finally, for the first assertion of the 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 the 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 of Lemma 5. □

3 Two-Party Computation with Randomness Preparation

In this section, first we recall in Section 3.1 the notion of secure two-party computation in the semi-honest model (in the "simulate the view" formulation, rather than another equivalent "ideal vs. real" formulation). Except some notational modifications, our formulation here is essentially based on the standard definitions described, e.g., in Section 7.2 of [7] and in [13]. Then in Section 3.2 we formalize, or clarify the meaning of, security (against a semi-honest party) of a two-party protocol in a practically possible situation where some party uses the random output of a certain algorithm (for example, a cryptographic pseudorandom number generator) for his/her random choices during the protocol instead of using theoretically ideal coin tosses.

3.1 The Semi-Honest Model

Let π be a two-party computation protocol between two 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 content of the 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 with inputs x_1, x_2 and random tapes r_1, r_2 is denoted by $\mathsf{out}_i^\pi(1^\lambda, x_1, x_2; r_1, r_2)$, or by $\mathsf{out}_i^\pi(1^\lambda, \vec{x}; \vec{r})$ in short where $\vec{x} := (x_1, x_2)$ and $\vec{r} := (r_1, r_2)$. We define $\mathsf{out}^\pi(1^\lambda, \vec{x}; \vec{r})$ to be the pair of $\mathsf{out}_1^\pi(1^\lambda, \vec{x}; \vec{r})$ and $\mathsf{out}_2^\pi(1^\lambda, \vec{x}; \vec{r})$ in this order. 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 positive 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.

In this paper, we let the *transcript* for Party $\mathcal{P}_i$ in an execution of π mean the sequence $(m_1^{(i)}, m_2^{(i)}, \ldots, m_{\ell_i}^{(i)})$ of messages (in the chronological order) sent to $\mathcal{P}_i$ from the other party 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 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 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.

From now, we describe the definition of security for a protocol π in the semi-honest model (it is often called with different names in the literature; e.g., "π privately computes f" in Section 7.2 of [7]; and "π securely computes f in the presence of static semi-honest adversaries" in [13]). Intuitively, the condition below means that the view for any of the two parties in an execution of π can be efficiently recovered, in a computationally indistinguishable manner (see Section 2.2), 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. See Section 2.1 for some notations.

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

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

We say that this simulator $\mathcal{S}_i$ is *statistically indistinguishable*, if the random variable families at the left-hand and the right-hand 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 $i \in \{1,2\}$.

3.2 Definition of Security against Randomness Prepared by a Party

Here we consider a practically possible situation in use of a two-party protocol π where, instead of a uniformly random bit sequence (as in the original situation), any of the two parties, say $\mathcal{P}_i$, uses the random output of a certain algorithm R_i as the content of his/her random tape in an execution of π. We call the R_i a *randomness preparation algorithm* for $\mathcal{P}_i$.

More precisely, let $\rho_i(\lambda)$ be (as in Section 3.1) the length of the random tape for Party $\mathcal{P}_i$ in an execution of π with security parameter 1^λ. First we give a technical definition:

Definition 4 (Admissible randomness preparation)**.** In the current setting, we say that an algorithm R_i is an *admissible* randomness preparation algorithm for Party $\mathcal{P}_i$, if the following conditions are all satisfied: The input for R_i consists of 1^λ and a possible input x_i for $\mathcal{P}_i$ with security parameter 1^λ; R_i is PPT with respect to λ; R_i outputs a $\rho_i(\lambda)$-bit sequence; and the internal randomness for R_i is an element of $\{0,1\}^{\widetilde{\rho}_i(\lambda)}$ for some positive polynomial $\widetilde{\rho}_i$.

It is worth emphasizing that the output of a randomness preparation algorithm for a party may be dependent on the input for the party in the current execution of the protocol. We also note that the algorithm, denoted by id, that simply outputs the content of its random tape is always admissible in this sense, where $\widetilde{\rho}_i = \rho_i$. Intuitively, the function id corresponds to the original case where the party simply uses the uniformly random tape rather than artificially generating it.

Now for $i \in \{1,2\}$, let R_i be an admissible randomness preparation algorithm for $\mathcal{P}_i$ as in Definition 4. Then, to study the aforementioned case where $\mathcal{P}_i$ uses a random output of R_i instead of the uniformly random tape, here we regard the situation as another two-party protocol between $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\pi \circ_i R_i$:

Definition 5 (Two-party protocol with randomness preparation)**.** Let $i \in \{1,2\}$, and let π and R_i be as above; in particular, the algorithm R_i is admissible in the sense of Definition 4. Then we define a two-party protocol $\pi \circ_i R_i$ as follows, where the new input set and the new random tape for the other party $\mathcal{P}_{3-i}$ are the same as π, while the new random tape for the party $\mathcal{P}_i$ has $\widetilde{\rho}_i(\lambda)$-bit length when the security parameter is λ:

1. Given a security parameter 1^λ, a local input x_i and a uniformly random tape $\widetilde{r}_i \leftarrow_R \{0,1\}^{\widetilde{\rho}_i(\lambda)}$, $\mathcal{P}_i$ first runs $R_i(1^\lambda, x_i; \widetilde{r}_i)$ with internal randomness $\widetilde{r}_i$, and obtains its output $r_i \in \{0,1\}^{\rho_i(\lambda)}$.

2. Then, given a local input x_{3-i} for the other party $\mathcal{P}_{3-i}$ further, the two parties $\mathcal{P}_1$ and $\mathcal{P}_2$ jointly execute the protocol π with input pair (x_1, x_2) and security parameter 1^λ, where $\mathcal{P}_i$ uses the r_i generated above as his/her random tape and $\mathcal{P}_{3-i}$ uses a uniformly random $r_{3-i} \leftarrow_R \{0,1\}^{\rho_{3-i}(\lambda)}$ as his/her random tape.

Here we emphasize that, the view for $\mathcal{P}_i$ in the new protocol $\pi \circ_i R_i$ involves the new random tape $\widetilde{r}_i$ rather than the prepared random tape r_i for the original π; but we note also that the r_i used during the protocol execution can be deterministically recovered from $\widetilde{r}_i$ and the party's input both of that are included in the view for the party.

Then we formalize "security for a protocol against a party's randomness preparation" in terms of the security (in the usual sense) for the modified protocol introduced in Definition 5:

Definition 6 (Security against randomness preparation)**.** Let π be a two-party protocol, and let $f = (f_1, f_2)$ be a functionality to be computed by the protocol π. Let $i \in \{1, 2\}$, and let $\mathcal{R}_i$ be a set of algorithms. We say that the protocol π (computing f) is *robust against randomness preparation class $\mathcal{R}_i$ by Party $\mathcal{P}_i$ under the semi-honest model*, if for any algorithm $R_i \in \mathcal{R}_i$ that is admissible for $\mathcal{P}_i$ with respect to the protocol π (see Definition 4 for the terminology), the protocol $\pi \circ_i R_i$ securely computes f against semi-honest $\mathcal{P}_i$ in the sense of Definition 3. We also use the terminology "against randomness preparation R_i" instead of "against randomness preparation class $\mathcal{R}_i$" above if $\mathcal{R}_i$ consists of a single (admissible) algorithm R_i only.

We note that the design of Definition 6 supposes the definition to be applied only to a protocol π that is secure against semi-honest $\mathcal{P}_i$ (without any randomness preparation) and the definition itself does not include the assumption on the security of the original π. Nevertheless, Definition 6 will also imply by itself the security of the original π (against semi-honest $\mathcal{P}_i$) if the set $\mathcal{R}_i$ of randomness preparation algorithms involves the "trivial" algorithm id mentioned above.

4 *Insecurity* against Indistinguishable Randomness Preparation

In this section, we provide a concrete example of a pair of a two-party protocol π (Section 4.1) and an (admissible) randomness preparation algorithm $\mathcal{F}$ (Section 4.2) for the first party $\mathcal{P}_1$, that causes the following "paradoxical" phenomenon:

- The protocol π securely computes a certain functionality f against semi-honest $\mathcal{P}_1$ (without randomness preparation) with even statistically indistinguishable simulator (without any computational assumption). In fact, our protocol π is not only secure against $\mathcal{P}_1$, but also securely computes f against semi-honest $\mathcal{P}_2$ as well.
- With uniformly random tape for $\mathcal{F}$, the output distribution of $\mathcal{F}$ is *statistically indistinguishable* from the uniformly random tape for $\mathcal{P}_1$ in π.
- However, π is *NOT robust against randomness preparation $\mathcal{F}$ by $\mathcal{P}_1$* under the semi-honest model (in the sense of Definition 6). Intuitively, this means that the security of one party $\mathcal{P}_2$ can be compromised by replacing the random tape of *the other party* $\mathcal{P}_1$ in an even *statistically indistinguishable* manner. (We note that, the randomness preparation by $\mathcal{P}_1$ does not affect the correctness of the protocol to compute the functionality f.)

4.1 The Secure Protocol

Here we construct the aforementioned secure two-party protocol π as follows (see Section 2.4 for definitions and properties about the Paillier cryptosystem used below):

- For a security parameter λ, the input for Party $\mathcal{P}_1$ is an integer $N \in [0, 2^{2\lambda} - 1]$; the input for Party $\mathcal{P}_2$ is a pair of two λ-bit integers p, q; and the functionality $f = (f_1, f_2)$ is defined by

$$(f_1(N,p,q), f_2(N,p,q)) := \begin{cases} (\perp, \perp) & \text{unless } p,q \text{ are primes, } p \neq q, \text{ and } p \equiv_4 q \equiv_4 3 \ , \\ (\perp', \perp') & \text{if } p,q \text{ are primes, } p \neq q, p \equiv_4 q \equiv_4 3, \text{ and } N \neq pq \ , \\ (\top, \top) & \text{if } p,q \text{ are primes, } p \neq q, p \equiv_4 q \equiv_4 3, \text{ and } N = pq \ . \end{cases}$$

1. $\mathcal{P}_2$ checks if p, q are primes, $p \neq q$, and $p \equiv_4 q \equiv_4 3$, by using any deterministic polynomial-time algorithm for primality test (for example, the AKS algorithm [1]). In case where some of the conditions fails, $\mathcal{P}_2$ sends $\perp$ to $\mathcal{P}_1$ and stops the protocol execution, and now both parties output $\perp$. Otherwise, $\mathcal{P}_2$ sends $\top$ to $\mathcal{P}_1$.

2. $\mathcal{P}_1$ generates a key pair $(\mathsf{pk}, \mathsf{sk})$ of the Paillier cryptosystem with plaintext space $\mathbb{Z}/N'\mathbb{Z}$, where $N' = p'q'$ and both of the prime factors of N' have at least $(2\lambda + 1)$-bit length. Then $\mathcal{P}_1$ generates a random ciphertext $[[N]]$ of N, and sends pk and the $[[N]]$ to $\mathcal{P}_2$.

3. $\mathcal{P}_2$ performs the following:

   (a) $\mathcal{P}_2$ generates a ciphertext $[[-pq]]$ of $-pq$, and generates $[[N - pq]]$ by applying the additively homomorphic operation to the $[[N]]$ and the $[[-pq]]$.

   (b) $\mathcal{P}_2$ computes $a = r \bmod N'$ by using a random $(3\lambda + 1)$-bit sequence r, and if $a \notin (\mathbb{Z}/N'\mathbb{Z})^\times$ then $\mathcal{P}_2$ replaces a with 1.

   (c) $\mathcal{P}_2$ generates $[[a(N-pq)]]$ from the $[[N-pq]]$ by applying the additively homomorphic operations combined with the standard double-and-add technique for scalar multiplication (hence the total number of operations is of order $O(\lambda)$).

   (d) $\mathcal{P}_2$ applies the statistically perfect re-randomization to the $[[a(N - pq)]]$, and sends the result to $\mathcal{P}_1$, which we also write $[[a(N - pq)]]$ by abusing notation.

4. $\mathcal{P}_1$ decrypts the given $[[a(N-pq)]]$ and obtains $a(N-pq) \in \mathbb{Z}/N'\mathbb{Z}$. In case where $a(N-pq) \neq 0$ in $\mathbb{Z}/N'\mathbb{Z}$, $\mathcal{P}_1$ sends $\perp'$ to $\mathcal{P}_2$ and stops the protocol execution, and now both parties output $\perp'$. Otherwise, $\mathcal{P}_1$ computes $y := r' \bmod N$ by using a random 3λ-bit sequence r', and if $y \notin (\mathbb{Z}/N\mathbb{Z})^\times$, then $\mathcal{P}_1$ replaces y with 1. Then $\mathcal{P}_1$ sends N and y to $\mathcal{P}_2$.

5. $\mathcal{P}_2$ computes $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ by using the algorithm of Lemma 4 in Section 2.5 where p and q play the role of prime factors of N. Then $\mathcal{P}_2$ sends x to $\mathcal{P}_1$ if $x^2 = y$ in $\mathbb{Z}/N\mathbb{Z}$, and sends $\perp''$ to $\mathcal{P}_1$ otherwise. Now the protocol halts, and both parties output $\top$.

First we verify that this protocol computes the functionality f correctly. The assertion holds obviously if the protocol stops at Step 1; from now, we suppose that the protocol has not stopped at Step 1. Secondly, at Step 4, we always have $a \in (\mathbb{Z}/N'\mathbb{Z})^\times$. On the other hand, $|N - pq|$ is less than both p' and q' by the choice of bit lengths of p' and q', therefore we have $N - pq \in (\mathbb{Z}/N'\mathbb{Z})^\times$ unless $N = pq$. This implies that, we have $a(N-pq) \neq 0$ if and only if $N \neq pq$. Hence the assertion of this paragraph holds.

From now, we prove the security of the protocol π in the semi-honest model, which is divided into the following two theorems.

Theorem 1. *The protocol π securely computes f against semi-honest $\mathcal{P}_1$ where the simulator is statistically indistinguishable.*

Proof. We construct a simulator $\mathcal{S}_1$ for $\mathcal{P}_1$. In the case where f_1 outputs $\perp$, $\mathcal{S}_1$ has to simply output a uniformly random tape for $\mathcal{P}_1$ and the transcript $(\perp)$, and it is obvious that the simulation is now perfect.

When f_1 outputs $\perp'$, first $\mathcal{S}_1$ generates the uniformly random tape for $\mathcal{P}_1$, and generates a ke pair $(\mathsf{pk}, \mathsf{sk})$ as in Step 2. Now, conditioned on the choice of $(\mathsf{pk}, \mathsf{sk})$, Lemmas 1 and 3 imply that the distribution of a chosen in Step 3 is statistically close to the uniform distribution on $(\mathbb{Z}/N'\mathbb{Z})^\times$ (with bound dependent solely on λ). On the other hand, since the output $\perp'$ of f_1 implies $N \neq pq$, we always have $N - pq \in (\mathbb{Z}/N'\mathbb{Z})^\times$ as in the argument above. Then it follows that $a(N - pq)$ is also statistically close to the uniform distribution on $(\mathbb{Z}/N'\mathbb{Z})^\times$ that is independent of the choices of N, p, and q. Moreover, owing to the statistically perfect re-randomization at Step 3d, in a real execution of π, $\mathcal{P}_1$ will receive at the end of Step 3 a ciphertext that is statistically indistinguishable (conditioned on any choice of the keys) from a uniformly random ciphertext of a uniformly random element of $(\mathbb{Z}/N'\mathbb{Z})^\times$. According to the observation, the simulator $\mathcal{S}_1$ then samples (in a statistically indistinguishable manner, owing to Lemmas 1 and 3 again) a uniformly random ciphertext c of a uniformly random element of $(\mathbb{Z}/N'\mathbb{Z})^\times$, and outputs the random tape for $\mathcal{P}_1$ chosen above and a transcript $(\top, c)$. This simulation is statistically indistinguishable by the argument above.

Finally, we consider the case where f_1 outputs $\top$; this case guarantees that $N = pq$ satisfies the condition for the Rabin function (see Section 2.5). First we note that, by an argument similar to the previous paragraph, in a real execution of π, $\mathcal{P}_1$ will receive at the end of Step 3 a ciphertext that is statistically indistinguishable (conditioned on any choice of the keys) from a uniformly random ciphertext of plaintext 0. Therefore, the simulator $\mathcal{S}_1$ can output *the part of* the random tape for $\mathcal{P}_1$ *used until the end of Step 3* and a transcript $(\top, [[0]])$ until the end of Step 3 in a similar, statistically indistinguishable manner. From now on, we concentrate on simulating the other part of the view relevant to Steps 4 and 5.

First we focus on the computation $y = r' \bmod N$ in a part of Step 4. In a real execution of π, the distribution of r' conditioned on a chosen y is uniform over the set $\{k \in [0, 2^{3\lambda} - 1] \mid k \bmod N = y\}$. Now Lemma 2 implies that the output distribution of a probabilistic function $g(y) := y + (u \bmod K_y) \cdot N$, where $K_y := \lfloor (2^{3\lambda} - 1 - y)/N \rfloor + 1$, with a uniformly random bit sequence u of sufficiently large (but polynomially bounded in and dependent solely on λ) length is statistically indistinguishable from the conditional distribution of r' (with bound dependent solely on λ).

On the other hand, in a real execution of π, the element y chosen in Step 4 is statistically close to the uniform distribution on $(\mathbb{Z}/N\mathbb{Z})^\times$ (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 5, denoted here by η, in the real execution of π is a uniformly random square root x of y in $\mathbb{Z}/N\mathbb{Z}$ if $y \in \mathsf{QR}_N$, and it is always $\perp''$ if $y \notin \mathsf{QR}_N$. Now let (x', y') denote an output of the algorithm in Lemma 5 (recall that this algorithm does not use knowledge of prime factors of N), and let η' denote an element computed in the same way as η but by using (x', y') instead of (x, y). Then by Lemma 5, we have $(x', y') \stackrel{s}{\equiv} (x, y)$, therefore $(r', x, y, \eta) \stackrel{s}{\equiv} (g(y), x, y, \eta) \stackrel{s}{\equiv} (g(y'), x', y', \eta')$. According to these arguments, the simulator $\mathcal{S}_1$ can also output (within polynomial time) $g(y')$ as the remaining part of the random tape for $\mathcal{P}_1$ and η' as the final part of the transcript, and the simulation for (r', η) is statistically indistinguishable by the argument above. This completes the proof of Theorem 1. □

Theorem 2. *Assume that the Paillier cryptosystem is CPA secure against non-uniform adversaries. Then the protocol π securely computes f against semi-honest $\mathcal{P}_2$.*

Proof. We construct a simulator $\mathcal{S}_2$ for $\mathcal{P}_2$. First we note that, $\mathcal{S}_2$ can simulate the behavior of $\mathcal{P}_2$ at Step 1 perfectly. In particular, $\mathcal{S}_2$ can perfectly simulate the view for $\mathcal{P}_2$ when f_2 outputs $\perp$.

We also note that, when the output of f_2 is different from $\perp$, $\mathcal{S}_2$ can also perfectly simulate the message received by $\mathcal{P}_2$ during Step 4 in a way independent of the behavior of π until the end of Step 3, given the output of f_2 and the input $(1^\lambda, p, q)$ for $\mathcal{P}_2$. Indeed, if f_2 outputs $\perp'$ then the received message is simply $\perp'$; while if f_2 outputs $\top$, then it guarantees that the input N for the other party is equal to pq, which is known by $\mathcal{S}_2$.

By the argument above, we may concentrate on simulating the message received by $\mathcal{P}_2$ during Step 2, denoted here by (pk', η), as well as the random tape for $\mathcal{P}_2$. Now note that, the distribution of (pk', η) is independent of the random tape for $\mathcal{P}_2$. Therefore, it suffices for $\mathcal{S}_2$ to simply output a uniformly random bit sequence as the perfectly simulated random tape. Hence, the task is now reduced to simulate (pk', η) by using given 1^λ, p, q, and the output of f_2 (different from $\perp$).

The simulation for (pk', η) by $\mathcal{S}_2$ is performed as follows: Generate a uniformly random bit sequence used in the key generation algorithm; generate a key pair $(\mathsf{pk}, \mathsf{sk})$ for the Paillier cryptosystem by using the chosen randomness; generate a random ciphertext $[[0]]$ of plaintext 0; and finally output $(\mathsf{pk}', \eta) := (\mathsf{pk}, [[0]])$.

Now assume, for the contrary, that a non-uniform PPT algorithm $\mathcal{D}$ can distinguish the family of the simulation results $(p, q; \mathsf{pk}, [[0]]; \nu)_{\lambda,N,p,q}$ and the family of the real views $(p, q; \mathsf{pk}, [[N]]; \nu)_{\lambda,N,p,q}$ where we write $\nu := (f_1(N, p, q), f_2(N, p, q))$. More precisely, there exists a positive polynomial poly satisfying the following: For any $n \geq 1$, there exist a $\lambda = \lambda_n > n$, advice z_λ associated to λ, and an input pair $(N_\lambda, (p_\lambda, q_\lambda))$, satisfying

$$\left|\Pr[\mathcal{D}^{(z_\lambda)}(1^\lambda, p_\lambda, q_\lambda, \mathsf{pk}, [[0]], \nu_\lambda) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(1^\lambda, p_\lambda, q_\lambda, \mathsf{pk}, [[N_\lambda]], \nu_\lambda) = 1]\right| \geq 1/\mathsf{poly}(\lambda)$$

where we write $\nu_\lambda := (f_1(N_\lambda, p_\lambda, q_\lambda), f_2(N_\lambda, p_\lambda, q_\lambda))$. Then we construct a non-uniform PPT adversary $\mathcal{A}$ against CPA security of the Paillier cryptosystem. We focus on the security parameters λ of the form $\lambda = \lambda_n$ for some n only, as the other cases are not relevant. The advice for $\mathcal{A}$ consists of N_λ, p_λ, q_λ, and z_λ. Given 1^λ and pk as inputs (we note that the choice of $\mathsf{pk} = N'$ is independent of the choice of N and (p, q)), $\mathcal{A}$ first decides the value of ν_λ depending on whether $N_\lambda = p_\lambda q_\lambda$ or not. Then $\mathcal{A}$ sends $m_0 := 0$ and $m_1 := N_\lambda$ to the challenger of the security game. On receiving the challenge ciphertext $[[m_\beta]]$ of plaintext m_β with $\beta \leftarrow_R \{0, 1\}$, $\mathcal{A}$ runs $\mathcal{D}^{(z_\lambda)}(1^\lambda, p_\lambda, q_\lambda, \mathsf{pk}, [[m_\beta]], \nu_\lambda)$ and then outputs the obtained output of $\mathcal{D}$. Now the input distribution for the $\mathcal{D}$ is identical to the one for the simulated situation and the real situation above if $\beta = 0$ and $\beta = 1$, respectively. Therefore, by the assumption on $\mathcal{D}$ above, the probabilities that $\mathcal{A}$ outputs 1 conditioned on the two choices of β have difference at least $1/\mathsf{poly}(\lambda)$. This means that $\mathcal{A}$ breaks the CPA security for the Paillier cryptosystem (against non-uniform adversary), which contradicts the assumption in the statement. Hence, $\mathcal{S}_2$ is a computationally indistinguishable simulator for $\mathcal{P}_2$. This completes the proof of Theorem 2. □

4.2 The Randomness Preparation Algorithm

Here we give an example of the aforementioned randomness preparation algorithm $\mathcal{F}$ for Party $\mathcal{P}_1$ in the two-party protocol π in Section 4.1, with the following two properties: The output of $\mathcal{F}$ (with uniformly random tape) is statistically indistinguishable from the original random tape for $\mathcal{P}_1$; and the protocol $\pi \circ_1 \mathcal{F}$ is *insecure against semi-honest* $\mathcal{P}_1$ (though π itself is secure as shown in Theorem 1). The construction of $\mathcal{F}$ is as follows:

- Given security parameter 1^λ and an input N for Party $\mathcal{P}_1$ in π as well as an internal random tape $\widetilde{r}$, $\mathcal{F}$ runs $\mathcal{S}_1(1^\lambda, N, \top; \widetilde{r})$ where $\mathcal{S}_1$ is the simulator for $\mathcal{P}_1$ constructed in Theorem 1, obtains its output (N, r, trans), and then outputs $\mathcal{F}(1^\lambda, N; \widetilde{r}) := r$.

Note that the algorithm $\mathcal{F}$ is admissible in the sense of Definition 4. Since the simulator $\mathcal{S}_1$ for $\mathcal{P}_1$ is statistically indistinguishable by Theorem 1, the output of $\mathcal{F}$ is also statistically indistinguishable from the original random tape for $\mathcal{P}_1$.

From now, we show that the protocol $\pi \circ_1 \mathcal{F}$ is indeed *insecure* against semi-honest $\mathcal{P}_1$. The next theorem 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 $\pi \circ_1 \mathcal{F}$, which was certainly concealed by the original protocol π:

Theorem 3. *For any input pair $(N, (p, q))$ for the two parties $\mathcal{P}_1, \mathcal{P}_2$ satisfying $f_1(N, (p, q)) = \top$, the Party $\mathcal{P}_1$ can efficiently determine the secret input (p, q) for $\mathcal{P}_2$ with probability at least $1/16 - \mathsf{negl}(\lambda)$ by using the view for $\mathcal{P}_1$ in the protocol $\pi \circ_1 \mathcal{F}$.*

Proof. First of all, the output $\top$ of f_1 guarantees that $N = pq$ is a composite integer as for the Rabin function (see Section 2.5).

We note that, the elements output by $\mathcal{S}_1$ during an execution of $\pi \circ_1 \mathcal{F}$ can be deterministically recovered from the random tape $\widetilde{r}_1$ for $\mathcal{P}_1$ in $\pi \circ_1 \mathcal{F}$. Now the output of $\mathcal{S}_1$, including a simulated random tape r'_1 for $\mathcal{P}_1$ in the original protocol π and a simulated message η (either an element $x' \in (\mathbb{Z}/N\mathbb{Z})^\times$ or $\perp''$) received in Step 5 of 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 4 is an element of QR_N with probability at least $1/4 - \mathsf{negl}(\lambda)$ for some negligible negl and, provided $y \in \mathsf{QR}_N$, the message received by $\mathcal{P}_1$ at Step 5 is an element x with $x^2 = y$. This implies that, for the corresponding element y' computed from r'_1 in $\pi \circ_1 \mathcal{F}$, 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 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 4 of π internally executed by $\pi \circ_1 \mathcal{F}$, 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 *set* $\{p, q\}$ of two prime factors of N, and then $\mathcal{P}_1$ can guess the *order* of the pair (p, q) as 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 $(1/4 - \mathsf{negl}(\lambda)) \cdot 1/2 \cdot 1/2 = 1/16 - \mathsf{negl}(\lambda)$ for some negligible negl. This completes the proof of Theorem 3. □

Given only the local input N (and the supposed local output $\top$), $\mathcal{P}_1$ can know that the input for $\mathcal{P}_2$ is a pair of prime factors of N, but it is still computationally hard to specify the actual input for $\mathcal{P}_2$ (unless factorizing the N is easy). On the other hand, by Theorem 3, the messages sent from $\mathcal{P}_2$ during the protocol $\pi \circ_1 \mathcal{F}$ enable $\mathcal{P}_1$ to determine the actual input for $\mathcal{P}_2$ with almost constant probability. This means that the protocol $\pi \circ_1 \mathcal{F}$ should never be regarded as secure against the $\mathcal{P}_1$, even though the protocol is obtained just by replacing the random tape in a protocol π secure against such $\mathcal{P}_1$ in a statistically indistinguishable manner.

We note that the protocol $\pi \circ_1 \mathcal{F}$ does indeed not satisfy the security requirement in Definition 3 (unless the integer factoring is easy). This may be intuitively obvious by the argument above, but we give a formal proof for the sake of completeness.

Theorem 4. *If the protocol $\pi \circ_1 \mathcal{F}$ above securely computes the f against semi-honest $\mathcal{P}_1$, then there exists a PPT algorithm that factorizes any integer $N = pq$ of the form in Section 2.5 with probability at least $1/8 - \mathsf{negl}(\lambda)$.*

Proof. Let $\widetilde{\mathcal{S}}_1$ be the PPT simulator for Party $\mathcal{P}_1$ yielded by the assumption on the security of $\pi \circ_1 \mathcal{F}$. We consider the following algorithm $\mathcal{A}$: Given 1^λ and N as inputs, $\mathcal{A}$ first runs $\widetilde{\mathcal{S}}_1(1^\lambda, N, \top)$ and obtains $(N, \overline{r}_1, \overline{\mathsf{trans}})$ where $\overline{\mathsf{trans}} := (\top, \overline{c}, \overline{x})$. Secondly, $\mathcal{A}$ runs $\mathcal{S}_1(1^\lambda, N, \top; \overline{r}_1)$ and obtains $(N, r_1'', \mathsf{trans}'')$ where $\mathsf{trans}'' := (\top, c'', x'')$. 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 $(1^\lambda, N, \widehat{r}_1, \widehat{\mathsf{trans}}, (\top, \top))$ as input where $\widehat{\mathsf{trans}} := (\top, \widehat{c}, \widehat{x})$, $\mathcal{D}$ runs $\mathcal{S}_1(1^\lambda, N, \top; \widehat{r}_1)$ and obtains $(N, \widehat{r}_1', \widehat{\mathsf{trans}}_1')$ where $\widehat{\mathsf{trans}}_1' := (\top, \widehat{c}_1', \widehat{x}')$; $\mathcal{D}$ computes $\widehat{y} \in (\mathbb{Z}/N\mathbb{Z})^\times$ as in Step 4 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 $\widehat{x} \notin \{\widehat{x}', -\widehat{x}'\}$, and outputs 0 otherwise. 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 $\pi \circ_1 \mathcal{F}$, the probability that $\mathcal{D}(1^\lambda, N, \overline{r}_1, \overline{\mathsf{trans}}, (\top, \top))$ outputs 1 has only negligible difference from the probability that $\mathcal{D}(1^\lambda, N, \widetilde{r}_1, (\top, c, x), (\top, \top))$ outputs 1, where $\widetilde{r}_1$ is a uniformly random tape for $\mathcal{P}_1$ in $\pi \circ_1 \mathcal{F}$ and $(\top, c, x)$ is the transcript for $\mathcal{P}_1$ in an execution of $\pi \circ_1 \mathcal{F}$ corresponding to the random tape $\widetilde{r}_1$.

For an execution of $\pi \circ_1 \mathcal{F}$, put $\mathcal{S}_1(1^\lambda, N, \top; \widetilde{r}_1) =: (N, r_1', (\top, c', x'))$, and let y' be the element of $(\mathbb{Z}/N\mathbb{Z})^\times$ computed as in Step 4 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 $1/4 - \mathsf{negl}(\lambda)$. Moreover, conditioned on these y' and x', the construction of $\pi \circ_1 \mathcal{F}$ implies that, for the corresponding elements y and x appeared in an execution of $\pi \circ_1 \mathcal{F}$ 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, $\mathcal{D}(1^\lambda, N, \widetilde{r}_1, (\top, c, x), (\top, \top))$ outputs 1 with probability at least $(1/4 - \mathsf{negl}(\lambda)) \cdot 1/2 = 1/8 - \mathsf{negl}(\lambda)$ for some negligible negl.

This implies that, the probability that $\mathcal{D}(1^\lambda, N, \overline{r}_1, \overline{\mathsf{trans}}, (\top, \top))$ outputs 1 is also at least $1/8 - \mathsf{negl}(\lambda)$ for some negligible negl. Moreover, by the construction of $\mathcal{D}$, the fact that the output of $\mathcal{D}(1^\lambda, N, \overline{r}_1, \overline{\mathsf{trans}}, (\top, \top))$ is 1 guarantees that $\mathcal{A}$ does not abort and correctly outputs the two factors of N (by the same argument as the proof of Theorem 3). This completes the proof of Theorem 4. □

5 Discussion on Existing Alternative Security Models

In this section, we revisit some existing security models for two-party computation, each of which is wished to be an alternative to the semi-honest model from some viewpoint. Here we deal with three existing models: the *augmented semi-honest model* introduced by Definition 7.4.24 in Section 7.4.4.1 of Goldreich's book [7] (Section 5.2); a restrictive variant of the augmented semi-honest model mentioned in a short note by Hazay and Lindell [8] (Section 5.3); and the *semi-malicious model* introduced by Asharov, Jain, and Wichs in Section A.2 of [2] (Section 5.4). As formulations of those models require the "ideal vs. real" framework for security definition instead of the simulator-based framework for the semi-honest security in Definition 3, we first recall the framework in Section 5.1 before going into details of each existing model.

In Sections 5.2 and 5.3 we show that, even in the augmented semi-honest model and the Hazay–Lindell's variant, the protocol π given in Section 4.1 (more precisely, a protocol obtained by modifying π in a "minimal" way to fit in those different models while keeping the essential property of π) is still secure against possibly malicious first party $\mathcal{P}_1$; while, as shown in Section 4.2, the protocol falls into insecure against semi-honest $\mathcal{P}_1$ when $\mathcal{P}_1$ utilizes the statistically indistinguishable randomness preparation algorithm $\mathcal{F}$ given there. This means that the problem which we

pointed out in Section 4 is not resolved by moving to those alternative models. (To be fair, we should emphasize that the original reasons of introducing those models were *not* focusing on some problem caused by the treatment of randomness in the two-party computation model as we did in Section 4; therefore it is not surprising that those models do not resolve the problem in Section 4.)

In contrast, in Section 5.4, we give a positive result that shows that, if a two-party protocol is secure against possibly malicious $\mathcal{P}_1$ under the semi-malicious model, then the protocol is also secure against semi-honest $\mathcal{P}_1$ even when $\mathcal{P}_1$ utilizes any randomness preparation algorithm; therefore, our problem in Section 4 can be resolved by adopting the semi-malicious model instead of the semi-honest model. This is in fact natural, since the definition of the semi-malicious model has already captured by itself (even implicitly) the robustness requirement against randomness preparation. We however note that, in comparison to our current problem which is only the modification of a party's random tape prior to the protocol execution, the semi-malicious model allows moreover a party to, roughly speaking, *adaptively* modify the party's *input* as well as the random tape (and also to abort the protocol execution) *at any timing* during a protocol execution provided the consistency of the party's behaviors until this timing can be defended with witness, which seems to be significantly stronger ability than just the prior randomness modification. By this reason, it would be not very convincing, or be too strict, to adopt the semi-malicious model as "the minimal requirement for practical two-party computation protocols" in the same way as the semi-honest model being (or seeming to be) widely regarded so at the present time.

5.1 Preliminaries: Ideal-Real Formulation of Security

In this section, let π denote any two-party protocol (rather than concrete protocols given in Section 4) computing some functionality $f = (f_1, f_2)$ between two parties $\mathcal{P}_1$ and $\mathcal{P}_2$. Here we recall the "ideal vs. real" formulation of security for π against possibly malicious parties; except slight differences of expressions, our description here essentially follows the standard formulation described, e.g., in Section 7.2 of [7]. We also note that, to simplify the description, here we concentrate on the security definition against the first party $\mathcal{P}_1$ only; i.e., the second party $\mathcal{P}_2$ is always assumed to be honest, since it is the case relevant to the problem shown in Section 4.

Below we let M mean some security model for two-party computation (e.g., the semi-honest model). First we describe the formulation of the "real case":

Definition 7 (Real execution of protocol)**.** Let $\mathcal{A}_1$ be any PPT algorithm (with interaction with some other algorithm). We say that $\mathcal{A}_1$ is an *admissible real strategy* for the first party $\mathcal{P}_1$ in a protocol π under a model M, if it specifies a possible behavior of $\mathcal{P}_1$ during an execution of π that is allowed under the model M. Then a real execution of π with first party's strategy $\mathcal{A}_1$ goes as follows:

- $\mathcal{P}_1$ is given a security parameter 1^λ, the local input x_1, and an auxiliary input z (chosen from a certain set associated to the parameter λ). On the other hand, $\mathcal{P}_2$ is given 1^λ and the local input x_2. (We note that, $\mathcal{P}_2$ is also given the same auxiliary input z under the formulation in Section 7.2 of [7], but here we omit it since the honest $\mathcal{P}_2$ must ignore the auxiliary input.)

- The two parties jointly execute the protocol π, where $\mathcal{P}_1$ behaves as specified by $\mathcal{A}_1$ and $\mathcal{P}_2$ behaves precisely as specified by π, and then obtain their own views during the protocol execution. We note that $\mathcal{P}_1$ may intentionally abort at any step of the execution provided it is specified so by $\mathcal{A}_1$ and is allowed under the model M.

- Then $\mathcal{P}_1$ outputs any information (which e.g., may be not the value of the function f_1 and may involve a guess for the input for $\mathcal{P}_2$ provided it is allowed under the model M) inferred

from his/her view in the way specified by $\mathcal{A}_1$. On the other hand, $\mathcal{P}_2$ simply outputs the value of the function f_2 that is implied by his/her view.

We define $\mathsf{REAL}_{\mathcal{A}_1,z}(1^\lambda, x_1, x_2) = \mathsf{REAL}^{\pi}_{\mathcal{A}_1,z}(1^\lambda, x_1, x_2)$ to be the pair of the two parties' outputs after the execution of π as above.

Secondly, we describe the formulation of the "ideal case":

Definition 8 (Ideal computation of functionality)**.** Let $\mathcal{B}_1$ be any PPT algorithm (with interaction with some other algorithm) that specifies a possible behavior of the first party $\mathcal{P}_1$ in an ideal computation of a functionality f explained below. We say that $\mathcal{B}_1$ is an *admissible ideal strategy* for $\mathcal{P}_1$ to compute f under a model M, if the behavior of $\mathcal{P}_1$ specified by $\mathcal{B}_1$ is allowed under the model M. Then an ideal computation of f with first party's strategy $\mathcal{B}_1$ goes as follows:

- $\mathcal{P}_1$ is given a security parameter 1^λ, the local input x_1, and an auxiliary input z. $\mathcal{P}_2$ is given 1^λ and the local input x_2.
- $\mathcal{P}_1$ either sends a modified input x_1' (which may be the x_1 itself) to a trusted third party $\mathcal{T}$, or decides to abort at the step, according to the strategy $\mathcal{B}_1$ (provided it is allowed under the model M). In any case, $\mathcal{P}_2$ simply sends the x_2 to $\mathcal{T}$.
- Unless $\mathcal{P}_1$ has aborted, $\mathcal{T}$ sends $f_1(x_1', x_2)$ to $\mathcal{P}_1$. Then $\mathcal{P}_1$ decides, according to the strategy $\mathcal{B}_1$, if he/she aborts at the step or not.
- If $\mathcal{P}_1$ has not aborted, then $\mathcal{T}$ sends $f_2(x_1', x_2)$ to $\mathcal{P}_2$. Otherwise, $\mathcal{T}$ only tells $\mathcal{P}_2$ that $\mathcal{P}_1$ has already aborted.
- Finally, regardless of whether $\mathcal{P}_1$ has aborted or not, $\mathcal{P}_1$ outputs any information inferred from his/her view (including any information received from $\mathcal{T}$) in the way specified by $\mathcal{B}_1$. On the other hand, $\mathcal{P}_2$ simply outputs the value received from $\mathcal{T}$ at the previous step if $\mathcal{P}_1$ has not aborted; otherwise $\mathcal{P}_2$ outputs abort.

We define $\mathsf{IDEAL}_{\mathcal{B}_1,z}(1^\lambda, x_1, x_2) = \mathsf{IDEAL}^{f}_{\mathcal{B}_1,z}(1^\lambda, x_1, x_2)$ to be the pair of the two parties' outputs after the process above.

Now the security against possibly malicious $\mathcal{P}_1$ is defined as follows:

Definition 9 (Security in ideal vs. real formulation)**.** Let π be a two-party protocol for functionality f. We say that π *securely computes* f *against* $\mathcal{P}_1$ *under a model* M, if for any PPT admissible real strategy $\mathcal{A}_1$ for $\mathcal{P}_1$ in π under the model M (see Definition 7), there exists a PPT admissible ideal strategy $\mathcal{B}_1$ for $\mathcal{P}_1$ to compute f under the model M (see Definition 8) satisfying

$$\left(\mathsf{IDEAL}^{f}_{\mathcal{B}_1,z}(1^\lambda, x_1, x_2)\right)_{\lambda,(x_1,x_2),z} \stackrel{c}{\equiv} \left(\mathsf{REAL}^{\pi}_{\mathcal{A}_1,z}(1^\lambda, x_1, x_2)\right)_{\lambda,(x_1,x_2),z} .$$

Moreover, when the two distributions above are statistically indistinguishable, we also say that π *computes* f *in a statistically close to ideal manner against* $\mathcal{P}_1$ *under a model* M.

We note that, when M is the semi-honest model, it is known that the security against $\mathcal{P}_1$ in the sense of Definition 9 is equivalent to the security against $\mathcal{P}_1$ in the sense of Definition 3; see e.g., Proposition 7.2.3 in [7].

5.2 The Augmented Semi-Honest Model

Here we consider the augmented semi-honest model introduced by Definition 7.4.24 in Section 7.4.4.1 of [7]. Except slight expressional differences, the definition of the model is as follows:

Definition 10 (Augmented semi-honest model)**.** We define the *augmented semi-honest model* to be the model where, in addition to the behaviors under the semi-honest model, a party is also allowed to perform as follows:

- Before starting the protocol execution, the party may either exchange the original local input for a modified input chosen from the same input set (and execute the protocol according to the modified input), or abort the execution.

  To be more precise, the party decides how to modify the input and whether he/she aborts or not, by using the security parameter 1^λ, the original local input x_i, the auxiliary input (if any), and a "special random tape" that is independent of the random tape for executing the protocol itself. Moreover, the final output by the party (in the real execution of the protocol) in the case where the party aborts at this timing is also independent of the part of the random tape for protocol execution.

- At any step where the party is supposed to send a message to the other party, the party may abort at the step without sending the message. To be more precise again, whether the party aborts or not depends on the special (independent) random tape as well as the party's currently obtained view.

Then, we would like to show that the protocol π in Section 4.1 is still secure against $\mathcal{P}_1$ under the augmented semi-honest model. However, there is an issue that, an augmented semi-honest $\mathcal{P}_1$ may now abort before the end of the protocol execution, while the original protocol π does not specify how an honest $\mathcal{P}_2$ should behave when $\mathcal{P}_1$ aborts. Here we slightly modify the protocol π to resolve this issue, in a simple way that an honest $\mathcal{P}_2$ will output abort whenever a possibly malicious $\mathcal{P}_1$ aborts before the end of the protocol. The resulting protocol is denoted by π^{aug}. We note that this modification does not affect the behavior of a semi-honest (or honest) party during the protocol; accordingly, the protocol $\pi^{\mathsf{aug}} \circ_1 \mathcal{F}$ (with $\mathcal{F}$ as in Section 4.2) is *insecure* against semi-honest $\mathcal{P}_1$ by the same reason as the case of $\pi \circ_1 \mathcal{F}$.

Then we have the following result, which, together with the aforementioned insecurity of $\pi^{\mathsf{aug}} \circ_1 \mathcal{F}$, means (as mentioned at the beginning of Section 5) that the problem in Section 4 is not resolved even by adopting the augmented semi-honest model:

Theorem 5. *The protocol π^{aug} above computes the same functionality f as π in a statistically close to ideal manner against $\mathcal{P}_1$ under the augmented semi-honest model (see Definition 9 for the terminology).*

Proof. Let $\mathcal{A}_1$ be any PPT admissible real strategy for $\mathcal{P}_1$ in π^{aug} under the augmented semi-honest model. Then we construct a PPT admissible ideal strategy $\mathcal{B}_1$ for $\mathcal{P}_1$ to compute f in the following manner. The random tape for $\mathcal{B}_1$ consists of the random tape $\widetilde{r}_1$ for the simulator $\mathcal{S}_1$ constructed in Theorem 1 and the special random tape $\widetilde{s}_1$ for $\mathcal{A}_1$ to be used for the input modification and for the abort decision. Given 1^λ, N, an auxiliary input z (if any), and a random tape $(\widetilde{r}_1, \widetilde{s}_1)$, the strategy $\mathcal{B}_1$ is defined as follows:

1. $\mathcal{B}_1$ runs $\mathcal{A}_1(1^\lambda, N, z; (*, \widetilde{s}_1))$ and obtains from the $\mathcal{A}_1$ the possibly modified input N^{aug} and the decision for whether the party aborts before the protocol execution or not, as well as the party's output in the case where the party aborts at this timing, where $*$ denotes some

content of the part of the random tape for $\mathcal{A}_1$ used in the protocol execution. Note that the aforementioned behavior of $\mathcal{A}_1$ observed by $\mathcal{B}_1$ at this step does not depend on the part $*$ of the random tape. Now, if the $\mathcal{A}_1$ decided to abort before the protocol execution, then $\mathcal{B}_1$ also decides to abort without sending the (modified) input to the trusted third party $\mathcal{T}$, and outputs what $\mathcal{B}_1$ was given by the $\mathcal{A}_1$ as the final output.

2. $\mathcal{B}_1$ sends N^{aug} to $\mathcal{T}$, and then receives an element η from $\mathcal{T}$.
3. $\mathcal{B}_1$ runs $\mathcal{S}_1(1^\lambda, N^{\mathsf{aug}}, (\eta, \eta); \widetilde{r}_1)$ and obtains its output $(N^{\mathsf{aug}}, \overline{r}_1, \overline{\mathsf{trans}})$.
4. $\mathcal{B}_1$ runs $\mathcal{A}_1(1^\lambda, N, z; (\overline{r}_1, \widetilde{s}_1))$ to obtain the following information; here, for each step during the protocol execution where some message should be sent from $\mathcal{P}_2$ to $\mathcal{P}_1$, $\mathcal{B}_1$ uses the corresponding part of the simulated transcript $\overline{\mathsf{trans}}$ for $\mathcal{P}_1$ as the message sent at the step.
   - $\mathcal{B}_1$ observes whether or not the $\mathcal{A}_1$ aborts before the end of the protocol execution. If the $\mathcal{A}_1$ aborted, then $\mathcal{B}_1$ decides to abort at this step (accordingly, $\mathcal{T}$ does not send the value of f_2 to $\mathcal{P}_2$). Otherwise, $\mathcal{B}_1$ does not abort (accordingly, $\mathcal{T}$ sends the value of f_2 to $\mathcal{P}_2$).
   - $\mathcal{B}_1$ obtains the final output by the $\mathcal{A}_1$, which is chosen as the final output by $\mathcal{B}_1$.

We note that, by the remark given in Step 1 above, in the real execution of the protocol π^{aug} between $\mathcal{A}_1$ and $\mathcal{B}_1$ (playing the role of $\mathcal{P}_2$) at Step 4 above, the $\mathcal{A}_1$ does not abort before the protocol execution and uses the same modified input N^{aug} as in Step 1. Note also that the $\mathcal{B}_1$ above is indeed admissible under the augmented semi-honest model.

We assume, for the contrary, that the two distributions $\left(\mathsf{IDEAL}^f_{\mathcal{B}_1,z}(1^\lambda, N, (p,q))\right)_{\lambda,(N,(p,q)),z}$ and $\left(\mathsf{REAL}^{\pi^{\mathsf{aug}}}_{\mathcal{A}_1,z}(1^\lambda, N, (p,q))\right)_{\lambda,(N,(p,q)),z}$ are not statistically indistinguishable. This means that, there exist a positive polynomial poly and infinite sequences of $\lambda = \lambda_n > n$ $(n = 1, 2, \dots)$, $N = N_\lambda$, $p = p_\lambda$, $q = q_\lambda$, and $z = z_\lambda$, satisfying that $\mathsf{IDEAL}^f_{\mathcal{B}_1,z_\lambda}(1^\lambda, N_\lambda, (p_\lambda, q_\lambda))$ and $\mathsf{REAL}^{\pi^{\mathsf{aug}}}_{\mathcal{A}_1,z_\lambda}(1^\lambda, N_\lambda, (p_\lambda, q_\lambda))$ have statistical distance at least $1/\mathsf{poly}(\lambda)$ for any of those $\lambda = \lambda_n$. This also implies the existence of a sequence $\widetilde{s}_{1,\lambda}$ of a content of the special random tape for which these two distributions *conditioned on the fixed choice of* $\widetilde{s}_{1,\lambda}$ also have statistical distance at least $1/\mathsf{poly}(\lambda)$ for any such λ. From now on, we fix the choice of $\widetilde{s}_{1,\lambda}$ and make it implicit in our argument. Now we note that $\mathcal{A}_1(1^\lambda, N_\lambda, z_\lambda)$ never aborts before the protocol execution; this is because, if it aborts, then $\mathcal{B}_1(1^\lambda, N_\lambda, z_\lambda)$ also aborts by the construction and now the two distributions above become identical. We also note that the $\mathcal{B}_1$ uses the same modified input $N^{\mathsf{aug}} = N^{\mathsf{aug}}_\lambda$ as the $\mathcal{A}_1$, and in the ideal case, the $\mathcal{B}_1$ receives $\eta = \eta_\lambda := f_1(N^{\mathsf{aug}}_\lambda, (p_\lambda, q_\lambda)) = f_2(N^{\mathsf{aug}}_\lambda, (p_\lambda, q_\lambda))$ from $\mathcal{T}$.

From now, we show that $\mathcal{S}_1(1^\lambda, N^{\mathsf{aug}}_\lambda, (\eta_\lambda, \eta_\lambda))$ has statistical distance at least $1/\mathsf{poly}(\lambda)$ from the real view for $\mathcal{P}_1$ in π with input N^{aug}_λ for $\mathcal{P}_1$ and input (p_λ, q_λ) for $\mathcal{P}_2$ for any such λ, which contradicts the statistical indistinguishability of $\mathcal{S}_1$ (Theorem 1) and hence will complete the proof. To show this, for each of such λ, we consider the following function $\mathcal{F}_\lambda$: Given $(N^{\mathsf{aug}}_\lambda, \overline{r}_1, \overline{\mathsf{trans}})$ as input, execute Step 4 in the construction of $\mathcal{B}_1$ with the 1^λ, $N = N_\lambda$ and $z = z_\lambda$ (and $\widetilde{s}_1 = \widetilde{s}_{1,\lambda}$), and output (ν_1, ν_2) where ν_1 is the final output of $\mathcal{B}_1$ and $\nu_2 := \mathsf{abort}$ if the $\mathcal{B}_1$ aborts during Step 4 and $\nu_2 := f_2(N^{\mathsf{aug}}_\lambda, (p_\lambda, q_\lambda))$ otherwise. Now if $(N^{\mathsf{aug}}_\lambda, \overline{r}_1, \overline{\mathsf{trans}}) = \mathcal{S}_1(1^\lambda, N^{\mathsf{aug}}_\lambda, (\eta_\lambda, \eta_\lambda))$, then the distribution of the value of $\mathcal{F}_\lambda$ is identical to the distribution of $\mathsf{IDEAL}^f_{\mathcal{B}_1,z_\lambda}(1^\lambda, N_\lambda, (p_\lambda, q_\lambda))$ by the construction of $\mathcal{B}_1$. On the other hand, if $(N^{\mathsf{aug}}_\lambda, \overline{r}_1, \overline{\mathsf{trans}})$ is a real view for $\mathcal{P}_1$ in the protocol π with input pair $(N^{\mathsf{aug}}_\lambda, (p_\lambda, q_\lambda))$, then the distribution of the value of $\mathcal{F}_\lambda$ is identical to the distribution of $\mathsf{REAL}^{\pi^{\mathsf{aug}}}_{\mathcal{B}_1,z_\lambda}(1^\lambda, N_\lambda, (p_\lambda, q_\lambda))$ by the construction of $\mathcal{B}_1$ again. Since the statistical distance of two distributions is not increased by applying a common function, it follows that the output distribution

of $\mathcal{S}_1(1^\lambda, N_\lambda^{\mathsf{aug}}, (\eta_\lambda, \eta_\lambda))$ has also statistical distance at least $1/\mathsf{poly}(\lambda)$ from the real view for $\mathcal{P}_1$ in π with input pair $(N_\lambda^{\mathsf{aug}}, (p_\lambda, q_\lambda))$, as desired. This completes the proof of Theorem 5. □

5.3 A Model in Hazay–Lindell's Note

Here we consider a model suggested in a short note by Hazay and Lindell [8]. In their model, the only ability of a party in addition to the semi-honest model is to modify the party's local input before the protocol execution (in the same sense as the augmented semi-honest model), while a party is not allowed to abort (in contrast to the augmented semi-honest model where a party may abort). Then we note that the situation for the model is the same as the case of the augmented semi-honest model in Section 5.2, as shown by the following result:

Theorem 6. *The protocol π^{aug} given in Section 5.2 computes the functionality f in a statistically close to ideal manner against $\mathcal{P}_1$ under the model above (see Definition 9 for the terminology).*

Proof. Owing to Theorem 5, the only task is to check that for any PPT admissible real strategy $\mathcal{A}_1$ for $\mathcal{P}_1$ in π^{aug} under the current model, the corresponding PPT ideal strategy $\mathcal{B}_1$ for $\mathcal{P}_1$ to compute f that was constructed in the proof of Theorem 5 is admissible under the current model. This is in fact obvious, as the $\mathcal{B}_1$ never aborts unless the $\mathcal{A}_1$ aborts by the construction. □

5.4 The Semi-Malicious Model

Here we consider the semi-malicious model introduced by Asharov, Jain, and Wichs in Section A.2 of [2]. Roughly speaking, a possibly malicious party in this model can do, at each step of a protocol, *anything provided the party can defend the consistency* of the current behavior and all the party's previous behaviors *by showing a witness* in a way as "I honestly used this input and this random tape!" *which may be adaptively forged* by the party. Although this model was introduced in [2] under Canetti's Universal Composability (UC) framework [3] which is different from our current setting, we can still formulate the essence of their model in a way fitting in our fashion, as follows:

Definition 11 (Semi-malicious model)**.** We define the *semi-malicious model* to be the model where a party's behavior in a two-party protocol π is specified as follows:

- At any timing during a protocol execution where the party is sending a message m to the other party, the party must also generate a *witness* of the form $(x^\dagger, r^\dagger)$ (which is not sent to the other party and is kept secret) with the following property: The sequence of all the actual messages previously sent to the other party and the current message m coincides with the sequence of expected messages that would have to be sent to the other party if the party were honestly following the protocol π, the party's local input were $x^\dagger$, the party's random tape were $r^\dagger$ and the party were receiving the same messages from the other party as the actually received ones.
- In addition, the party may abort a protocol execution at any timing, even before starting the protocol execution.

Here we give a remark on the formulation of the ideal computation under the semi-malicious model. In Definition 8, a possibly malicious Party $\mathcal{P}_1$ may abort during the computation process; this ability to abort during an ideal computation should also be allowed under the semi-malicious model by naturally interpreting the description above (or the description in [2]). On the other hand, a possibly malicious $\mathcal{P}_1$ in Definition 8 may also modify his/her input sent to the trusted third party $\mathcal{T}$. There seem to exist two possible options for this point, to or not to allow the

modification of the input sent to $\mathcal{T}$; it would depend on whether or not we should interpret "the input sent to $\mathcal{T}$ (in an ideal computation)" as a kind of "a message m to the other party" in the description above, which seems to be not very clear. In this paper, we choose an option to allow the input modification in an ideal computation by default, which then results in the same model of ideal computation as the case of the augmented semi-honest model (see Section 5). We also sometimes choose the other option not allowing the input modification, in which case it will be specified explicitly.

As the semi-malicious model already concerns the modification of the random tape (with "trivial" witness involving the modified random tape itself), it is naturally expected that any protocol secure in the semi-malicious model is also robust against randomness preparation (in contrast to the previous cases of the augmented semi-honest model and Hazay–Lindell's model, where the implication as above is denied by the protocol π^{aug}). However, we do not have a proof for this expected implication in a general case so far, due to some technical difficulty caused by the aforementioned ability of modifying the local input in an ideal computation of the functionality f; namely, even if a real strategy for the first party $\mathcal{P}_1$ is semi-honest, there remains the following potential possibility for the corresponding ideal strategy $\mathcal{B}_1$ for $\mathcal{P}_1$ to compute f under the semi-malicious model: $\mathcal{B}_1$ may be able to modify the local input x_1 to some other $x_1^\dagger$ in a way that $\mathcal{B}_1$ infers some non-trivial information on the other party's input x_2 from the value of $f_1(x_1^\dagger, x_2)$ received from the trusted third party that is different from $f_1(x_1, x_2)$ (hence not known by the simulator for $\mathcal{P}_1$) while keeping the output $f_2(x_1^\dagger, x_2)$ by the other party unchanged from $f_2(x_1, x_2)$ (being consistent with the indistinguishability between the real and the ideal cases). In the following results on the implication from semi-malicious security to robustness against randomness preparation, we avoid the technical difficulty in two ways; to restrict the functionality f to those for which the input modification ability above does not matter (Theorem 7); and to forbid the input modification in the ideal computation of f (Theorem 8). One may observe the aforementioned technical difficulty further by comparing the proofs of Theorems 7 and 8. We also note that, even the restricted Theorem 7 suffices to show that the protocol π^{aug} in Section 5.2 (nor the original protocol π in Section 4.1) cannot be a counterexample even if the expected implication above does in fact not hold.

From now, we give the two implication results mentioned above:

Theorem 7. *Let π be a two-party protocol, and let $f = (f_1, f_2)$ denote the functionality that is computed by π between two honest parties. Suppose that π securely computes f against $\mathcal{P}_1$ under the semi-malicious model. Suppose also that f satisfies one of the following conditions:*

- *$f_1 = f_2$ are identical and deterministic functions.*
- *The output of f_1 is independent of the input for $\mathcal{P}_1$.*

Then π is robust against any admissible randomness preparation R_1 by $\mathcal{P}_1$ under the semi-honest model. Moreover, if π computes f in a statistically close to ideal manner against $\mathcal{P}_1$ under the semi-malicious model, then the simulator for $\mathcal{P}_1$ in $\pi \circ_1 R_1$ is statistically indistinguishable.

Proof. Let R_1 be any admissible (in particular, PPT) randomness preparation algorithm for $\mathcal{P}_1$. Now we consider the following PPT real strategy $\mathcal{A}_1$ (without auxiliary input z) for $\mathcal{P}_1$ in π under the semi-malicious model: Given 1^λ, a local input x_1, and a uniformly random tape $\widetilde{r}_1$, $\mathcal{A}_1$ first runs $R_1(1^\lambda, x_1; \widetilde{r}_1)$ and obtains its output $r_1^\dagger$; secondly, $\mathcal{A}_1$ honestly executes the protocol π with input x_1 and random tape $r_1^\dagger$, where the witness required at each step is simply $(x_1, r_1^\dagger)$; and then outputs $(x_1, \widetilde{r}_1, \mathsf{trans}^\dagger)$ where $\mathsf{trans}^\dagger$ denotes the transcript received from $\mathcal{P}_2$ during the protocol execution of π above. Note that this $\mathcal{A}_1$ is indeed admissible under the semi-malicious model. By the assumption

on the security of π under the semi-malicious model, there exists a PPT admissible ideal strategy $\mathcal{B}_1$ for $\mathcal{P}_1$ to compute f under the semi-malicious model, for which the output distributions of the real execution (using $\mathcal{A}_1$) and the ideal computation (using $\mathcal{B}_1$) is computationally (or statistically, resp.) indistinguishable.

Then we construct a PPT simulator $\mathcal{S}_1$ for semi-honest $\mathcal{P}_1$ in the protocol $\pi \circ_1 R_1$ as follows:

1. Given 1^λ, x_1 and an element ν as inputs, $\mathcal{S}_1$ first runs $\mathcal{B}_1$ with input $(1^\lambda, x_1)$ and observes how $\mathcal{B}_1$ modifies the input to be sent to the trusted third party $\mathcal{T}$ and whether or not $\mathcal{B}_1$ decides to abort before sending an input to $\mathcal{T}$. If $\mathcal{B}_1$ decides to abort, then $\mathcal{S}_1$ outputs abort and halts. Otherwise, let $x_1^\dagger$ denote the (possibly) modified input to be sent to $\mathcal{T}$.

2. Secondly, $\mathcal{S}_1$ continues to run the $\mathcal{B}_1$ with the element ν playing the role of the element received from $\mathcal{T}$, and outputs what the $\mathcal{B}_1$ finally outputs.

We show that the output distribution of $\mathcal{S}_1$ is computationally (or statistically, resp.) indistinguishable from the view for semi-honest $\mathcal{P}_1$ in an execution of $\pi \circ_1 R_1$. First we note that, the output distribution of the real execution of π between $\mathcal{A}_1$ and honest $\mathcal{P}_2$ is, by construction of $\mathcal{A}_1$, identical to the pair of the view for $\mathcal{P}_1$ and the output by $\mathcal{P}_2$ in an execution of $\pi \circ_1 R_1$. Therefore, it suffices to show that the output distribution of $\mathcal{S}_1$ is statistically indistinguishable from the output distribution of $\mathcal{B}_1$ in the ideal computation of f with honest $\mathcal{P}_2$.

We note that $\mathcal{A}_1$ never aborts (hence $\mathcal{P}_2$ never outputs abort) in a real execution of π as an honest party never aborts. Therefore, in the ideal computation of f between $\mathcal{B}_1$ and $\mathcal{P}_2$, $\mathcal{B}_1$ aborts (hence $\mathcal{P}_2$ outputs abort) with at most negligible probability (depending solely on λ); otherwise, a distinguisher can easily distinguish the real execution and the ideal computation by just seeing the output by $\mathcal{P}_2$, a contradiction.

Here we consider the first case in the statement that $f_1 = f_2$ are identical and deterministic functions. As an honestly executed π computes f correctly by definition of f, and as f is now deterministic, an honestly executed protocol $\pi \circ_1 R_1$ also computes f correctly. It follows by construction of $\mathcal{A}_1$ that, $\mathcal{P}_2$ always outputs the correct value of $f_2(x_1, x_2)$ in a real execution of π with $\mathcal{A}_1$. Then, in the ideal computation of f between $\mathcal{B}_1$ and $\mathcal{P}_2$, the output by $\mathcal{P}_2$ is different from $f_2(x_1, x_2)$ with at most negligible probability depending solely on λ, since otherwise a distinguisher can easily distinguish the real execution and the ideal computation by just checking whether the output by $\mathcal{P}_2$ is equal to $f_2(x_1, x_2)$ or not, a contradiction again (we note that, since we are considering *non-uniform* distinguishers in the definition of indistinguishability, a distinguisher can learn the "best to distinguish" input pair (x_1, x_2) as auxiliary advice, and then the distinguisher can know the correct value of $f_2(x_1, x_2)$). Since the output by $\mathcal{P}_2$ in the ideal computation of f is $f_2(x_1^\dagger, x_2)$, and since f_1 is identical to f_2 by the assumption, it follows that $f_1(x_1^\dagger, x_2) = f_2(x_1^\dagger, x_2) = f_2(x_1, x_2) = f_1(x_1, x_2) = \nu$ except negligible probability. This implies that the output distribution of $\mathcal{B}_1$ executed internally in $\mathcal{S}_1$ (hence that of $\mathcal{S}_1$ itself) has only negligible distance from the output distribution of $\mathcal{B}_1$ in the ideal computation of f with honest $\mathcal{P}_2$, as desired.

Finally, we consider the other case in the statement where the output of f_1 is independent of the first input. In this case, the assumption implies that $f_1(x_1^\dagger, x_2)$ and $f_1(x_1, x_2) = \nu$ have identical probability distributions. Therefore, since $\mathcal{B}_1$ aborts with negligible probability as shown above, it follows that the output distribution of $\mathcal{B}_1$ executed internally in $\mathcal{S}_1$ (hence that of $\mathcal{S}_1$ itself) has only negligible distance from the output distribution of $\mathcal{B}_1$ in the ideal computation of f with honest $\mathcal{P}_2$, as desired. This completes the proof of Theorem 7. □

Theorem 8. *Let π be a two-party protocol, and let $f = (f_1, f_2)$ denote the functionality that is computed by π between two honest parties. Suppose that π securely computes f against $\mathcal{P}_1$ under the*

"optional" semi-malicious model where the input modification in the ideal computation of f is not allowed (see the remark after Definition 11). Then π is robust against any admissible randomness preparation R_1 by $\mathcal{P}_1$ under the semi-honest model. Moreover, if π computes f in a statistically close to ideal manner against $\mathcal{P}_1$ under the "optional" semi-malicious model, then the simulator for $\mathcal{P}_1$ in $\pi \circ_1 R_1$ is statistically indistinguishable.

Proof. The proof is basically the same as Theorem 7: R_1, $\mathcal{A}_1$, $\mathcal{B}_1$ and $\mathcal{S}_1$ are as in the proof of Theorem 7. Now the "optional" condition for the semi-malicious model implies (without any restriction on the functionality f as in the statement of Theorem 7) that, unless $\mathcal{B}_1$ aborts before sending an input to the trusted third party, we always have $x_1^\dagger = x_1$ and hence $f_1(x_1^\dagger, x_2)$ and $f_1(x_1, x_2) = \nu$ have identical probability distributions (where $x_1^\dagger$ and ν are as in the proof of Theorem 7). Since $\mathcal{B}_1$ aborts with at most negligible probability by the same reason as the case of Theorem 7, it follows that the output distribution of $\mathcal{B}_1$ executed internally in $\mathcal{S}_1$ (hence that of $\mathcal{S}_1$ itself) has only negligible distance from the output distribution of $\mathcal{B}_1$ in the ideal computation of f with honest $\mathcal{P}_2$. This completes the proof in the same way as Theorem 7. □

6 On Conditions to Prevent Randomness Preparation Problem

In this section, we give another positive result showing some sufficient conditions to prevent the problem (pointed out in Section 4) caused by a party's randomness preparation. In contrast to Theorems 7 and 8 in Section 5.4, our conditions can be fully checked within the semi-honest model, without introducing some stronger adversary models such as the semi-malicious model. Instead, we require a simulator constructed in a security proof for the original protocol under the semi-honest model to satisfy some additional conditions, and also suppose that the output of the randomness preparation algorithm is sufficiently indistinguishable from uniform (while, in Theorems 7 and 8 above, we put essentially no assumption on the randomness preparation algorithms). We note that these conditions are not always required to prove the robustness against randomness preparation for a specific protocol; we give an example of a protocol to see this below.

Our requirement for a simulator in the security proof for the original protocol mentioned above is the following:

Definition 12. 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, $\mathcal{S}_i$ generates a uniformly random tape r_i for $\mathcal{P}_i$, 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 then 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 a uniformly 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 Theorem 2 is in fact with raw random tape, while the simulator $\mathcal{S}_1$ in the proof of Theorem 1 is *not* with raw random tape.

From now, we give our aforementioned positive result on preventing the randomness preparation problem under some conditions. Here we say that a randomness preparation algorithm R_i for Party $\mathcal{P}_i$ is *computationally indistinguishable* (or *statistically indistinguishable*, resp.), if the output distribution of R_i with uniformly random tape is computationally (or statistically, resp.) indistinguishable from the uniformly random tape for $\mathcal{P}_i$. Then the result is as follows:

Theorem 9. *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 *(see Definition 12 for the terminology), then π is robust against any PPT admissible,* statistically indistinguishable *randomness preparation algorithm R_i (see above for the terminology) by $\mathcal{P}_i$ under the semi-honest model. Moreover, for any such R_i, $\pi \circ_i R_i$ 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 12. By definition, $(\mathcal{X}_{\lambda,\vec{x}})_{\lambda,\vec{x}} := \big(x_1, r_1, \mathcal{T}_1(1^\lambda, x_1, f_1(\vec{x}), r_1), f(\vec{x})\big)_{\lambda,\vec{x}}$ is statistically indistinguishable from $(\mathcal{Y}_{\lambda,\vec{x}})_{\lambda,\vec{x}} := \big(x_1, r_1, \mathsf{trans}_1^\pi(1^\lambda, \vec{x}; r_1, r_2), \mathsf{out}^\pi(1^\lambda, \vec{x}; r_1, r_2)\big)_{\lambda,\vec{x}}$, 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 $R_1(1^\lambda, x_1)$. By the assumption on R_1, $\overline{r}_1$ is statistically indistinguishable from r_1. 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 probability distribution of the random tape $\widetilde{r}_1$ for $R_1(1^\lambda, x_1)$ conditioned on the event $R_1(1^\lambda, x_1; \widetilde{r}_1) = \overline{r}_1$. 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 $R_1(1^\lambda, x_1; \widetilde{r}_1)$ 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 tape for $R_1(1^\lambda, x_1)$. Moreover, we always have $R_1(1^\lambda, x_1; s(\overline{r}_1)) = \overline{r}_1$ by the definition of $s(\overline{r}_1)$. Therefore, the random variable $\mathcal{X}''_{\lambda,\vec{x}}$ can be rewritten as

$$\mathcal{X}''_{\lambda,\vec{x}} = \big(x_1, s, \mathcal{T}_1(1^\lambda, x_1, f_1(\vec{x}), R_1(1^\lambda, x_1; s)), f(\vec{x})\big)$$

where the s follows the uniform distribution for the random tape for $R_1(1^\lambda, x_1)$, and the random variable $\mathcal{Y}''_{\lambda,\vec{x}}$ can be rewritten as

$$\mathcal{Y}''_{\lambda,\vec{x}} = \big(x_1, s, \mathsf{trans}_1^\pi(1^\lambda, \vec{x}; R_1(1^\lambda, x_1; s), r_2), \mathsf{out}^\pi(1^\lambda, \vec{x}; R_1(1^\lambda, x_1; s), r_2)\big)$$

where the s follows again the uniform distribution for the random tape for $R_1(1^\lambda, x_1)$. By the definition of $\pi \circ_1 R_1$, this $\mathcal{Y}''_{\lambda,\vec{x}}$ is also equal to

$$\mathcal{Y}''_{\lambda,\vec{x}} = \big(x_1, s, \mathsf{trans}_1^{\pi\circ_1 R_1}(1^\lambda, \vec{x}; s, r_2), \mathsf{out}^{\pi\circ_1 R_1}(1^\lambda, \vec{x}; s, r_2)\big) \ .$$

According to the argument above, we now define a simulator $\widetilde{S}_1$ for $\mathcal{P}_1$ in $\pi \circ_1 R_1$ 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 $\mathsf{trans}_1 := \mathcal{T}_1(1^\lambda, x_1, f_1(\vec{x}), R_1(1^\lambda, x_1; s))$, and then outputs $(x_1, s, \mathsf{trans}_1)$. This is a PPT algorithm as well as $\mathcal{T}_1$ and R_1, 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 $\pi\circ_1 R_1$ is statistically indistinguishable. This completes the proof of Theorem 9. □

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 indistinguishability" of the randomness preparation algorithm. First, among the three conditions, the necessity of the "raw random tape" property is clear, as the results in Section 4 imply that the protocol π and the randomness preparation algorithm $\mathcal{F}$ given there will be a counterexample for Theorem 9 (unless the integer factoring is easy) once the "raw randomness tape" assumption is removed from the statement.

Secondly, we discuss the importance of the "statistical" indistinguishability assumption on the simulator. To see this, suppose instead that the simulator is only computationally indistinguishable (while keeping the other two conditions). By the notations in the proof above, this means that the two families of distributions $(\mathcal{X}_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ and $(\mathcal{Y}_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ are computationally indistinguishable. Then the computational indistinguishability of $(\mathcal{X}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ and $(\mathcal{Y}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ can be similarly deduced from the computational (in fact, statistical) indistinguishability of the randomness preparation algorithm. 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 $(\mathcal{X}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ and $(\mathcal{Y}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ would imply in a similar way that $(\mathcal{X}''_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ and $(\mathcal{Y}''_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ are also computationally indistinguishable, which is the desired conclusion. However, in fact $s(\overline{r}_1)$ is *not* efficiently samplable in general (e.g., consider any one-way permutation on the set $\{0,1\}^{\rho(\lambda)}$ of random tapes, where the output distribution in forward direction is perfectly uniform as desired, but sampling the s means inverting the permutation which is computationally hard by definition), and our proof strategy fails at this point if the "statistical" indistinguishability assumption on the simulator is weakened to the computational one.

One may feel that, among the three conditions above, the "statistical indistinguishability" of the randomness preparation algorithm 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 indistinguishable (or "cryptographic") pseudorandom generator (PRG) to prepare the party's own random tape. Unfortunately, the use of such a PRG causes two hurdles in our proof strategy above. The first hurdle is the same as the previous paragraph, namely, it is generally hard to sample a seed of a PRG conditioned on a given output of the PRG, i.e., $s(\overline{r}_1)$ by the notation in the previous paragraph. In fact, this hurdle would be resolved *if $(\mathcal{X}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ and $(\mathcal{Y}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ were statistically indistinguishable*. But actually, it is only guaranteed that $(\mathcal{X}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ and $(\mathcal{Y}'_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ are *only computationally* indistinguishable even though $(\mathcal{X}_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ and $(\mathcal{Y}_{\lambda,\vec{x}})_{\lambda,\vec{x}}$ are assumed to be statistically indistinguishable, since the PRG used is only computationally indistinguishable. This is the second hurdle caused by using a PRG in the current situation.

We emphasize that, the necessity of the three conditions discussed above is for a *generic* proof strategy, and the robustness against randomness preparation can still be proved for a *specific* protocol even without (at least) one of the three conditions. We give an example of a two-party protocol to see this, which is an equality test of two parties' inputs based on the Paillier cryptosystem. This protocol in fact has already appeared as a part of the protocol π in Section 4.1, but here we describe the protocol again for the sake of completeness, denoted by $\pi_=$:

- The inputs x_1, x_2 are elements of a finite subset I_λ of non-negative integers. The functionality $f = (f_1, f_2)$ is defined by $f_1(\vec{x}) = f_2(\vec{x}) = 1$ if $x_1 = x_2$ and $f_1(\vec{x}) = f_2(\vec{x}) = 0$ if $x_1 \neq x_2$.

1. $\mathcal{P}_1$ generates a key pair $(\mathsf{pk}, \mathsf{sk})$ of the Paillier cryptosystem with plaintext space $\mathbb{Z}/N\mathbb{Z}$, where N is an $\eta(\lambda)$-bit integer for a positive polynomial $\eta(\lambda) \geq \lambda$, and both of the prime factors of N are larger than the maximum of I_λ (hence I_λ can be regarded as a subset of $\mathbb{Z}/N\mathbb{Z}$). Then $\mathcal{P}_1$ generates a random ciphertext $[[x_1]]$ of x_1, and sends pk and the $[[x_1]]$ to $\mathcal{P}_2$.

2. $\mathcal{P}_2$ generates a ciphertext $[[-x_2]]$ of $-x_2$, and generates $[[x_1 - x_2]]$ by applying the additively

homomorphic operation to the $[[x_1]]$ and the $[[-x_2]]$.

3. $\mathcal{P}_2$ computes $a = r \bmod N$ by using a random $(\eta(\lambda)+\lambda)$-bit sequence r, and if $a \notin (\mathbb{Z}/N\mathbb{Z})^\times$ then $\mathcal{P}_2$ replaces a with 1. Then $\mathcal{P}_2$ generates $[[a(x_1-x_2)]]$ from the $[[x_1-x_2]]$ by applying the additively homomorphic operations combined with the standard double-and-add technique for scalar multiplication (hence the total number of operations is of order $O(\eta(\lambda))$).

4. $\mathcal{P}_2$ applies the statistically perfect re-randomization to the $[[a(x_1-x_2)]]$, and sends the result to $\mathcal{P}_1$, which we also write $[[a(x_1-x_2)]]$ by abusing notation.

5. $\mathcal{P}_1$ decrypts the given $[[a(x_1-x_2)]]$ and obtains $a(x_1-x_2) \in \mathbb{Z}/N\mathbb{Z}$. Secondly, $\mathcal{P}_1$ sets $\chi := 1$ if $a(x_1-x_2) = 0$ in $\mathbb{Z}/N\mathbb{Z}$, and $\chi := 0$ otherwise. Then $\mathcal{P}_1$ sends χ to $\mathcal{P}_2$, and $\mathcal{P}_1$ and $\mathcal{P}_2$ output χ.

Theorem 10. *For any PPT admissible randomness preparation algorithm R_1 (including the case $R_1 = \mathsf{id}$) for the first party $\mathcal{P}_1$ in the protocol $\pi_=$, the protocol $\pi_= \circ_1 R_1$ securely computes the functionality f against semi-honest $\mathcal{P}_1$ where the simulator is statistically indistinguishable with raw random tape. On the other hand, if the Paillier cryptosystem is CPA secure against non-uniform adversaries, then for any PPT admissible randomness preparation algorithm R_2 (including the case $R_2 = \mathsf{id}$) for the second party $\mathcal{P}_2$ in $\pi_=$, the protocol $\pi_= \circ_2 R_2$ securely computes f against semi-honest $\mathcal{P}_2$ where the simulator is with raw random tape.*

Proof. First, we construct a simulator $\mathcal{S}_1$ for $\mathcal{P}_1$ in $\pi_= \circ_1 R_1$. Given 1^λ, x_1 and $\chi = f_1(\vec{x}) \in \{0,1\}$ as inputs, $\mathcal{S}_1$ first generates a uniformly random tape $\widetilde{r}_1$ for R_1, and runs $R_1(1^\lambda, x_1; \widetilde{r}_1)$ to obtain the random tape r_1 for $\mathcal{P}_1$ in $\pi_=$. Secondly, $\mathcal{S}_1$ chooses a key pair $(\mathsf{pk}, \mathsf{sk})$ as in the protocol by using the random tape r_1. Now if $\chi = 1$, $\mathcal{S}_1$ generates a random ciphertext c of plaintext 0. On the other hand, if $\chi = 0$, $\mathcal{S}_1$ generates a random ciphertext c of a (statistically close to) uniformly random plaintext chosen from $(\mathbb{Z}/N\mathbb{Z})^\times$ (by virtue of Lemmas 1 and 3). Finally, $\mathcal{S}_1$ outputs $(x_1, \widetilde{r}_1, c)$. This $\mathcal{S}_1$ is PPT and is with raw random tape by the construction. Note also that $\mathcal{S}_1$ perfectly simulates the first step of $\pi_= \circ_1 R_1$.

In the case $x_1 = x_2$, we have $a(x_1-x_2) = 0$, therefore the ciphertext $[[a(x_1-x_2)]]$ received at the fourth step of the protocol is statistically close to a uniformly random ciphertext of 0 (owing to the statistically perfect re-randomization). Hence the protocol correctly computes the functionality, and $\mathcal{S}_1$ simulates the transcript in a statistically indistinguishable manner in this case.

On the other hand, in the other case $x_1 \neq x_2$, the element $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ chosen in the third step of the protocol is statistically close to uniform (owing to Lemmas 1 and 3), while $x_1 - x_2 \in (\mathbb{Z}/N\mathbb{Z})^\times$ due to the choice of N in the first step (which guarantees that $|x_1 - x_2|$ is smaller than any of the two prime factors of N). This implies that $a(x_1 - x_2) \in (\mathbb{Z}/N\mathbb{Z})^\times$ (in particular $a(x_1 - x_2) \neq 0$) and it is also statistically close to uniform over $(\mathbb{Z}/N\mathbb{Z})^\times$. Hence the protocol correctly computes the functionality, and $\mathcal{S}_1$ simulates the transcript in a statistically indistinguishable manner in this case as well. Therefore, the assertion for security against $\mathcal{P}_1$ holds (we note that here we did not rely on the security assumption on the Paillier cryptosystem).

Secondly, we construct a simulator $\mathcal{S}_2$ for $\mathcal{P}_2$ in $\pi_= \circ_2 R_2$. Given 1^λ, x_2 and $\chi = f_2(\vec{x}) \in \{0,1\}$ as inputs, $\mathcal{S}_2$ first generates a uniformly random tape $\widetilde{r}_2$ for R_2, and runs $R_2(1^\lambda, x_2; \widetilde{r}_2)$ to obtain the random tape r_2 for $\mathcal{P}_2$ in $\pi_=$. Secondly, $\mathcal{S}_2$ generates the uniformly random tape r_1 for $\mathcal{P}_1$, and chooses a key pair $(\mathsf{pk}, \mathsf{sk})$ as in the protocol by using the random tape r_1. Thirdly, $\mathcal{S}_2$ generates a random ciphertext $[[0]]$ of plaintext 0. Finally, $\mathcal{S}_2$ outputs $(x_2, \widetilde{r}_2, (\mathsf{pk}, [[0]], \chi))$. This $\mathcal{S}_2$ is PPT and is with raw random tape by the construction. Note that the protocol correctly computes the functionality by the same argument as above; that is, the simulation for the part χ of the transcript is perfect. We note also that $\mathcal{S}_2$ perfectly simulates the part pk of the transcript as well.

Now assume, for the contrary, that a non-uniform PPT distinguisher $\mathcal{D}$ can distinguish the simulation result $\mathcal{X}_{\lambda,\vec{x}} := (x_2, \widetilde{r}_2, (\mathsf{pk}, [[0]], \chi), f(\vec{x}))$ and the real behavior $\mathcal{Y}_{\lambda,\vec{x}} := (x_2, \widetilde{r}_2, \mathsf{trans}_2, \mathsf{out})$ of $\pi_= \circ_2 R_2$. More precisely, there exists a positive polynomial poly satisfying the following: For any $n \geq 1$, there exist a $\lambda = \lambda_n > n$, an input pair $\vec{x}_\lambda = (x_{\lambda,1}, x_{\lambda,2})$, and advice $z = z_\lambda$ satisfying $\left|\Pr[\mathcal{D}^{(z_\lambda)}(1^\lambda, \mathcal{X}_{\lambda,\vec{x}_\lambda}) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(1^\lambda, \mathcal{Y}_{\lambda,\vec{x}_\lambda}) = 1]\right| \geq 1/\mathsf{poly}(\lambda)$. Then we construct a non-uniform PPT adversary $\mathcal{A}$ against the CPA security of the Paillier cryptosystem. Here we focus only on the security parameters of the form $\lambda = \lambda_n$ for some n as the other cases are not relevant. The advice for $\mathcal{A}$ consists of $\vec{x}_\lambda$ and z_λ. Given 1^λ and pk as inputs, $\mathcal{A}$ chooses a uniformly random tape $\widetilde{r}_2$ for R_2 (we note that the choice of pk in the protocol is independent of the random tape for $\mathcal{P}_2$), and sends two plaintexts $m_0 := 0$ and $m_1 := x_{\lambda,1}$ to the challenger. On receiving the challenge ciphertext $[[m_\beta]]$ of plaintext m_β with $\beta \leftarrow_R \{0,1\}$, $\mathcal{A}$ sets $\mathcal{Z} := (x_{\lambda,2}, \widetilde{r}_2, (\mathsf{pk}, [[m_\beta]], \chi), (\chi, \chi))$ where $\chi := f_1(x_{\lambda,1}, x_{\lambda,2}) = f_2(x_{\lambda,1}, x_{\lambda,2})$, runs $\mathcal{D}^{(z_\lambda)}(1^\lambda, \mathcal{Z})$, and then outputs the output of the $\mathcal{D}$. Now by the construction, the distribution of $\mathcal{Z}$ is identical to $\mathcal{X}_{\lambda,\vec{x}_\lambda}$ and to $\mathcal{Y}_{\lambda,\vec{x}_\lambda}$ if $\beta = 0$ and $\beta = 1$, respectively. Therefore, by the assumption on $\mathcal{D}$, the probabilities that the $\mathcal{A}$ outputs 1 conditioned on the two choices of β have difference at least $1/\mathsf{poly}(\lambda)$ as well, for any such λ. This means that $\mathcal{A}$ breaks the CPA security (against non-uniform adversary) of the Paillier cryptosystem, a contradiction. Hence the simulator $\mathcal{S}_2$ is computationally indistinguishable, as desired. This completes the proof of Theorem 10. □

Acknowledgments. The author thanks all the members of a study group "Shin-Akarui-Angou-Benkyoukai" for fruitful discussions on the results of this paper. In particular, the author is most grateful to Shota Yamada for his smart advice on this study. And the author also specially thanks the following members; Kazumasa Shinagawa, Takashi Yamakawa, Takahiro Matsuda, Yusuke Sakai, Keita Emura, and Goichiro Hanaoka, for their precious comments.

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] R. Canetti: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Proceedings of FOCS 2001, pp.136–145, 2001.

[4] 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.

[5] 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.

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

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

[8] C. Hazay, Y. Lindell: A Note on the Relation between the Definitions of Security for Semi-Honest and Malicious Adversaries. IACR Cryptology ePrint Archive 2010/551 (`http://eprint.iacr.org/2010/551`), 2010 (Version 20101101:164326).

[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.

[12] P. Hubácek, D. Wichs: On the Communication Complexity of Secure Function Evaluation with Long Output. IACR Cryptology ePrint Archive 2014/669 (`http://eprint.iacr.org/2014/669`), 2014 (Version 20140828:224736).

[13] 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).

[14] 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.

[15] 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).

[16] 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.

[17] P. Paillier: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Proceedings of EUROCRYPT 1999, LNCS vol.1592, pp.223–238, 1999.

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

[19] K. Shimizu, K. Nuida, H. Arai, S. Mitsunari, N. Attrapadung, M. Hamada, K. Tsuda, T. Hirokawa, J. Sakuma, G. Hanaoka, K. Asai: Privacy-Preserving Search for Chemical Compound Databases. BMC Bioinformatics, vol.16(Suppl 18), article no.S6, 2015.

[20] K. Shimizu, K. Nuida, G. Rätsch: Efficient Privacy-Preserving String Search and an Application in Genomics. Bioinformatics, vol.32, no.11, pp.1652–1661, 2016.

[21] Wikipedia: "Tool-Assisted Speedrun". `https://en.wikipedia.org/wiki/Tool-assisted_speedrun` (accessed on November 2, 2016).