Cryptographic Pseudorandom Generators Can Make Cryptosystems Problematic

Koji Nuida[1,2][0000−0001−8259−9958]

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

Abstract. Randomness is an essential resource for cryptography. For practical randomness generation, the security notion of pseudorandom generators (PRGs) intends to automatically preserve (computational) security of cryptosystems when used in implementation. Nevertheless, some opposite case such as in computational randomness extractors (Barak et al., CRYPTO 2011) is known (but not yet systematically studied so far) where the security can be lost even by applying secure PRGs. The present paper aims at pushing ahead the observation and understanding about such a phenomenon; we reveal such situations at layers of primitives and protocols as well, not just of building blocks like randomness extractors. We present three typical types of such cases: (1) adversaries can legally see the seed of the PRGs (including the case of randomness extractors); (2) the set of "bad" randomness may be not efficiently recognizable; (3) the formulation of a desired property implicitly involves non-uniform distinguishers for PRGs. We point out that the semi-honest security of multiparty computation also belongs to Type 1, while the correctness with negligible decryption error probability for public key encryption belongs to Types 2 and 3. We construct examples for each type where a secure PRG (against uniform distinguishers only, for Type 3) does not preserve the security/correctness of the original scheme; and discuss some countermeasures to avoid such an issue.

Keywords: Pseudorandom generators · public key encryption · multiparty computation

1 Introduction

Randomness is an essential resource for cryptography. While theoretical design of cryptosystems usually relies on ideal randomness, it is practically expensive to generate a large amount of (almost) ideal randomness, therefore some efficient "approximation" of randomness is necessary. When computational security is sufficient, a standard way is to use cryptographically secure pseudorandom generators (PRGs) in implementation. Due to the way of defining the security of PRGs (i.e., computational indistinguishability of the output from being uniformly random), it is widely expected in the area of cryptography that if the cryptosystem is secure assuming ideal randomness, and the PRG is also secure, then the cryptosystem implemented by the PRG instead of the ideal randomness will be secure as well. Indeed, usually no security caution is given when a cryptosystem is implemented by using a cryptographically secure PRG; such a use of PRG is even frequently recommended by professional cryptographers.

However, in fact there exists some situation where (computational) security of a cryptographic scheme is *not* preserved by implementation using a secure PRG. Namely, Barak et al. has shown in Section 4.1 of [3] the following. Let $\mathsf{Ext}(X;S)$ be a randomness extractor with source distribution X and random seeds chosen from S. We consider the situation that a random seed $s \leftarrow S$ is replaced by a PRG's output $\mathcal{R}(s_0)$ with shorter seed $s_0 \leftarrow S_0$. Roughly speaking, their result gives a pair of a secure extractor $\mathsf{Ext}(X;S)$ and a secure PRG $\mathcal{R}$ that yields an *insecure* extractor $\mathsf{Ext}(X;\mathcal{R}(S_0))$. A consequence is that the aforementioned standard methodology of implementing the randomness by secure PRGs does *not always* guarantee the security of the implemented scheme. (Some conditions to avoid such a loss of security are also discussed in their paper.) This fact should have impact for evaluating security of practically used cryptosystems where the use of cryptographic PRGs is recommended. Nevertheless, to the author's best knowledge, such a phenomenon caused by PRGs has not been systematically studied in the literature. The present paper aims at pushing ahead the observation and understanding about such a phenomenon for the case of other kinds of cryptographic schemes.

1.1 Our Contributions

In this paper, we look at the aforementioned possible phenomenon that some required property of (computationally secure) cryptographic schemes may be lost by applying PRGs even if the PRG itself is secure. We point out the following three types of typical situations where such a phenomenon may happen.

Type 1: The seed of the PRG is visible for adversaries.

This includes the known case of randomness extractors Ext mentioned above. Namely, its security is defined as $\mathsf{Ext}((X;S),S,Z) \overset{\mathsf{c}}{\approx} (U,S,Z)$ under certain conditions for X and Z where $\overset{\mathsf{c}}{\approx}$ denotes the computational indistinguishability and U denotes the uniform distribution on some set (see Definition 4 of [3] for details). The essence is that *the adversary in the security notion* (i.e., the distinguisher behind the notation $\overset{\mathsf{c}}{\approx}$) *can also see the internal randomness S of* Ext. On the other hand, the security definition $\mathcal{R}(S_0) \overset{\mathsf{c}}{\approx} U$ for a PRG $\mathcal{R}$ supposes that the seed (internal randomness) is not visible for the adversary. Intuitively, as the security of PRGs does not suppose the case where the internal randomness is visible for the adversary, the security of the PRG may be useless to preserve the security of the randomness extractor with visible seeds.

Here we point out that such a security notion with visible randomness in fact also appears in situations closer to real applications (rather than just building blocks like randomness extractors). Concretely, the standard security notion for *multiparty computation* (MPC) is also of this type (see Section 3.1 for details).

Here we focus on two-party computation (2PC) among MPC for the sake of simplicity, and give the following result.

Theorem 1 (Informal). *Under a certain assumption, there is a pair of a 2PC protocol π and a secure PRG $\mathcal{R}$ with the following property: π is secure (in the semi-honest model) against a party $\mathcal{P}$ but the protocol becomes insecure against the party $\mathcal{P}$ when the internal randomness for $\mathcal{P}$ is generated by using $\mathcal{R}$.*

See Section 3 for details. Roughly summarizing, we construct two pairs $(\pi_1, \mathcal{R}_1)$ and $(\pi_2, \mathcal{R}_2)$ as in the statement; π_1 is artificially constructed but is very simple; while π_2 is complicated but is a practical protocol chosen from a paper by Asharov et al. in ACM CCS 2013 [1] (more precisely, Protocol 51 in Section 5.2 of its full version [2]). We note that possibilities for such connections between a party's randomness and the security against the same party have been suggested in some previous papers [17, 22], but no concrete example of the connection was given in the literature before the present work. (We also note that the underlying assumption in the theorem is not a standard one, which is a main drawback of the result. Nevertheless, the assumption is at least not immediately falsifiable, which suggests that it would not be able to guarantee in general that a secure PRG preserves the security of MPC.)

It should be emphasized that there is no contradiction in the theorem where the semi-honest security is lost by applying a secure PRG, as the semi-honest model requests each party to follow the protocol *precisely, including the ideal randomness generation*. However, the possible gap between security of MPC with ideal randomness and with PRGs seems to be not recognized in the research area; our result here gives a caution for this point. In the author's opinion, the situation for (semi-honest) MPC with PRGs would have to be similar to cryptography in the random oracle model (ROM) where most of the cryptographers know the gap between ROM and the real (cf. Section 1.2 below) and they explicitly accept the rigorous imperfectness as a trade-off with practical efficiency.

We might expect that such a loss of security would not occur for "natural" cases, especially with "natural" PRGs, as the construction of PRG $\mathcal{R}$ in our theorem above is very artificial and impractical. But the meaning of "natural" here is not rigorous; it is worthy to establish some sufficient conditions for *provably* preventing such a loss of security. Towards this affirmative direction, in this paper we give the following result. Here we say (roughly) that a simulator $\mathcal{S}$ for a party $\mathcal{P}$ in a security proof of a 2PC protocol is *with raw randomness*, if $\mathcal{S}$ generates the simulated randomness for $\mathcal{P}$ by using a part of randomness for $\mathcal{S}$ "as is" (rather than adjusting according to the other part of the output of $\mathcal{S}$); see Definition 1 in Section 3.5 for the precise definition. We also recall that the *min-entropy* of a random variable X is defined by $H_\infty(X) = -\max_x \log_2 \Pr[X = x]$.

Theorem 2 (Informal). *Let π be a semi-honest 2PC protocol that is information-theoretically secure against a party $\mathcal{P}$ with raw randomness for simulator (see above for the terminology). Let $\mathcal{R}$ be a PRG and suppose that the difference of min-entropy of $\mathcal{R}$'s output distribution from that of ideal randomness is at most of logarithmic order (with respect to the security parameter). Then by generating* *the randomness for $\mathcal{P}$ with $\mathcal{R}$, the protocol π remains information-theoretically secure against semi-honest $\mathcal{P}$ with raw randomness for simulator.*

See Section 3.5 for details. We emphasize that if we remove the condition of "with raw randomness for simulator" (respectively, "information-theoretically secure") from the hypothesis, then the protocol-PRG pair $(\pi_1, \mathcal{R}_1)$ (respectively, $(\pi_2, \mathcal{R}_2)$) appeared in the proof of Theorem 1 gives a counterexample, therefore the condition is essential in the statement.

On the other hand, the current condition for PRG in the theorem (which implies that the PRG has only logarithmic stretch) looks very severe and it is important to weaken the condition. In particular, it is desirable for such a theorem to be based on some computational property of PRGs, rather than information-theoretic one such as min-entropy. Here we intuitively explain a difficulty behind the problem; let $\mathcal{S}$ and $\mathcal{S}_{\mathrm{PRG}}$ be simulators to be constructed in the security of an original protocol Π and its variant Π_{PRG} using a PRG $\mathcal{R}$, respectively. To show that the security of Π implies the security of Π_{PRG}, it suffices to show an implication from $\mathcal{S}$ to $\mathcal{S}_{\mathrm{PRG}}$, or equivalently, that if the output of $\mathcal{S}_{\mathrm{PRG}}$ can be distinguished by an algorithm D_{PRG} then the output of $\mathcal{S}$ will also be distinguished by some algorithm D. When constructing D from D_{PRG}, a straightforward strategy (using D_{PRG} in a black-box manner) would involve a process to convert a given input for D into an input for D_{PRG}. However, now an input for D involves randomness for Π (to be generated by $\mathcal{R}$ in the case of Π_{PRG}) and an input for D_{PRG} involves a seed for $\mathcal{R}$; hence, such a conversion as above might require a kind of "inversion" of $\mathcal{R}$ from its output to its seed, which would be difficult due to the security of $\mathcal{R}$. Our proof in this paper escapes successfully from such a difficulty in the reduction-based proof by utilizing the extremely high min-entropy for the PRG. It looks a challenging task to handle such a difficulty by basing on computational security of the PRG.

Type 2: The "bad" randomness may be not efficiently recognizable. Intuitively, when the security of some cryptosystem against a (polynomial-time) adversary (who cannot see the internal randomness) is concerned, it suffices for the PRG to fool this adversary only, therefore the usual security of the PRG can ensure that the security of the cryptosystem is preserved. In contrast, here we point out that the security of PRGs may be not sufficient to preserve the *correctness* of a cryptosystem; the security is of course important, but the correctness should be even more important. We focus only on the case of public key encryption (PKE); to point out the existence of such a phenomenon is a main purpose of the present work, and more exhaustive studies among other kinds of cryptographic schemes are future research topics.

When a PKE scheme has perfect (zero-error) correctness, the way of randomness generation does not affect the correctness at all. On the other hand, here we deal with PKE schemes with negligible but *non-zero* decryption error probability, and we want to generate the randomness for key generation by using a PRG. The issue we point out is the following: even if the ratio of "bad" randomness yielding a key with high error probability is negligible among the whole space, in general *the set of "bad" randomness may be not efficiently recognizable*[3]. If the set were efficiently recognizable, the security of a PRG would ensure that the probability of choosing "bad" randomness is only negligibly changed by the PRG, therefore the correctness would be preserved. But it is in general not true, therefore the probability of choosing "bad" randomness may increase non-negligibly even if the PRG is secure[4]:

Theorem 3 (Informal). *Under a certain assumption, there is a pair of a PKE scheme and a secure PRG with the following property: the probability of choosing "bad" randomness in the key generation is exponentially small when the ideal randomness is used but becomes* 1 *when the output of the PRG is used instead.*

See Section 4 for details. Such an issue of "bad" randomness may potentially occur also in other cryptosystems. Although the example in the theorem is artificially constructed and the author has not found any such example among the schemes proposed in the literature, the result still suggests that it might be important to check if the set of "bad" randomness is efficiently recognizable when designing a new cryptosystem; such an issue in correctness (rather than security) has not been noticed in the literature to the author's best knowledge.

We note that there is a general solution (at least for PKE) to avoid such an issue, which is a conversion method to make the scheme perfectly correct, proposed by Bitansky and Vaikuntanathan [5][5]. But the method has large overhead and is not very practical. The situation is similar also for the Type 3 below.

Type 3: Non-uniform distinguishers are implicitly related.

For example, the standard security notion for MPC (cf. Section 7.2 of [16]) is explicitly based on the indistinguishability of random variables against *non-uniform* distinguishers with advice $z = z_\lambda$ dependent solely on the security parameter λ. Then it is natural that the PRG should also be secure against non-uniform distinguishers. In contrast, here we point out that there are cases in cryptography where non-uniform security (not just the security against uniform distinguishers) is required for the PRG *but the relevance of non-uniformity is implicit*. Concretely, we again deal with the correctness with negligible errors for PKE, but here we focus on the encryption algorithm rather than key generation. To the author's best knowledge, such relevance of non-uniform security for PRGs to the *correctness*[6] of PKE has not been studied in the literature.

[3] "Performing key generation (using the randomness), encryption, and decryption and then checking if the result is correct" is in general *not* an efficient procedure, as the corresponding "bad" plaintext to be encrypted may be not efficiently samplable.

[4] The issue remains even if the PRG is secure against non-uniform distinguishers with advice. Although the set of "bad" randomness is fixed for each security parameter, this set may be too complicated to be included in the advice of polynomial length.

[5] Such so-called "immunization" methods had also been studied before, e.g., [13, 20, 23], but those methods remove the errors only partially. We note also that such methods did not concern the issue as in the paper and their motivations were different; e.g., preventing attacks that utilize decryption errors (e.g., [21]).

[6] For the *security* of PKE, the theory can be reasonably based on the uniform complexity treatment [14].

An intuitive explanation is as follows. In a usual definition for correctness, the decryption error probability has to be negligible for *any* plaintext. When falsifying the correctness (under the use of a PRG), the error probability will be non-negligible for *some* plaintext. The essence is that *such a "bad" plaintext m_λ at each security parameter λ is not necessarily found in polynomial time*, therefore a distinguisher for the PRG that utilizes the plaintexts m_λ should be non-uniform with advice m_λ. More precisely, we give the following result.

Theorem 4 (Informal). *Under a certain assumption (including the gap between uniform and non-uniform security for PRGs*[7]*), there is a pair of a PKE scheme and a (uniformly) secure PRG for which the decryption error probability is exponentially small when the ideal randomness is used in encryption but becomes non-negligible when the output of the PRG is used instead.*

See Section 5 for details. We note that any non-uniformly secure PRG used in the encryption algorithm preserves the correctness. But switching from uniform to non-uniform security may worsen the security parameter in practical implementations, due to some results on attacks by non-uniform algorithms, e.g., [4, 7, 26]. We also give a possible strategy of avoiding non-uniformly secure PRGs in ensuring the correctness after the use of a PRG; see Theorem 10 for details.

1.2 Related Work

One may feel some similarity of the results in this paper to a famous result by Canetti, Goldreich, and Halevi [6] showing that there is a scheme involving a (keyless) hash function that is provably secure when the hash function is modeled as a random oracle but becomes insecure for any concrete implementation of the hash function. In some sense, both of the present paper and theirs reveal gaps between cryptography based on idealized frameworks (ideal randomness / ROM) and that based on real objects (PRGs / hash functions). We emphasize, however, that there exists the following difference between the two results; the "real objects" in [6] (hash functions) themselves do not have provable security, while the present paper shows that even *provably secure* "real objects" (PRGs) can cause insecurity in implementation, which may have stronger impact. (On the other hand, a point of the present paper weaker than theirs is that our result here shows the existence of at least one "problematic" real object, while [6] shows that *any* such real object is "problematic".)

We also note another related result by Hirose [18] that for any (keyless) hash function under a certain model of construction that is secure when an ideal block cipher is used in the construction, there exists a block cipher that is provably secure but by which the resulting hash function becomes insecure. This result also focused on insecurity caused by provably secure building blocks, but our result in this paper covers wider situations, not just hash functions.

One may also feel that the topic of the present paper seems to be related to some other topics concerning non-ideal randomness in cryptography, such

[7] The issue discussed here will disappear if there is no such gap.

as cryptography based on so-called "imperfect randomness" (e.g., [10,12]) and the security issues caused by "backdoored PRGs" (e.g., [8,9]). But actually, the former topic above mainly deals with randomness that is significantly far from being ideal; in contrast, the present paper focuses on the use of randomness that is significantly close to ideal. On the other hand, the latter topic above studies the problem of the use of maliciously (and secretly) designed PRGs; while the main concern of the present paper originates from the practical impossibility of implementing the ideal randomness even if an engineer is honest and makes a best effort. Hence our problem setting is significantly different.

Finally, we mention about a previous work by Dodis et al. [11] which also studies situations where some internal states of a PRG are leaked to an adversary. An advantage of their result is that security notions for PRGs concerning such situations are established and precise constructions of PRGs satisfying their conditions are given. However, we emphasize that their security notion in fact considers only *partial* leakage of inputs to the PRG; in sequential updates of the internal state depending on newly supplied random seeds, an adversary obtains some intermediate states and then the PRG intends to quickly recover an unpredictable state with the help of subsequent *unknown* seeds. In contrast, Type 1 in our argument here considers more severe cases where the *entire* input (seed) to the PRG is known by an adversary; due to the difference of situations, the affirmative results in [11] would not (straightforwardly) resolve our problem.

2 Preliminaries

For a probabilistic algorithm $\mathcal{A}$, we may write $\mathcal{A}(x;r)$ instead of $\mathcal{A}(x)$ to emphasize the choice of randomness r. We adopt a convention that an advice $z = z_\lambda$ for a non-uniform algorithm $\mathcal{A} = \mathcal{A}^{(z_\lambda)}$ depends solely on the security parameter λ.[8] We let "polynomial-time" mean "polynomial-time with respect to λ". For a finite set S, let $\Delta(X,Y) = (1/2)\sum_{z \in S} |\Pr[z \leftarrow X] - \Pr[z \leftarrow Y]|$ be the statistical distance of random variables X and Y on S. Let $U[S]$ denote the uniform distribution on S. We write $x \leftarrow_R S$ to mean that x is sampled from S uniformly at random. We may identify a bit sequence with an integer via binary expressions of integers.

Let I_λ ($\lambda \geq 1$) be index sets. Let $X = (X_{\lambda,w})_{\lambda,w}$ and $Y = (Y_{\lambda,w})_{\lambda,w}$ be families of random variables indexed by $\lambda \geq 1$ and $w \in I_\lambda$. We say that X and Y are *uniformly* (respectively, *non-uniformly*) *indistinguishable*, denoted by $X \overset{\mathsf{u.c}}{\approx} Y$ (respectively, $X \overset{\mathsf{nu.c}}{\approx} Y$), if for any probabilistic polynomial-time (PPT) uniform (respectively, non-uniform) distinguisher $\mathcal{D}$, there is a negligible function $\varepsilon(\lambda) \in \lambda^{-\omega(1)}$ satisfying that the *advantage* $|\Pr[\mathcal{D}(1^\lambda, X_{\lambda,w}) = 1] - \Pr[\mathcal{D}(1^\lambda, Y_{\lambda,w}) = 1]|$ is at most $\varepsilon(\lambda)$ for any λ and $w \in I_\lambda$. We say that X and Y are *information-theoretically indistinguishable*, denoted by $X \overset{\mathsf{i}}{\approx} Y$, if there is a negligible function $\varepsilon(\lambda)$ with $\Delta(X_{\lambda,w}, Y_{\lambda,w}) \leq \varepsilon(\lambda)$ for any λ and $w \in I_\lambda$.

[8] By an appropriate padding to the input, our convention here can be made consistent with a standard convention where an advice depends solely on the input length.

In this paper, we let a *pseudorandom generator* (*PRG*) $\mathcal{R}$ be a deterministic polynomial-time algorithm that takes security parameter 1^λ and a seed $s \in \{0,1\}^{\ell_{\mathrm{in}}(\lambda)}$ as input and outputs an element of $\{0,1\}^{\ell_{\mathrm{out}}(\lambda)}$, where $\ell_{\mathrm{in}}(\lambda)$ and $\ell_{\mathrm{out}}(\lambda)$ are some polynomially bounded and polynomial-time computable functions satisfying that $\lambda \leq \ell_{\mathrm{in}}(\lambda) < \ell_{\mathrm{out}}(\lambda)$ and $\ell_{\mathrm{in}}(\lambda)$ is a strictly increasing function[9]. We say that a PRG $\mathcal{R}$ is *uniformly* (respectively, *non-uniformly*) *secure*, if $\mathcal{R}(1^\lambda, U[\{0,1\}^{\ell_{\mathrm{in}}(\lambda)}]) \overset{\mathsf{u.c}}{\approx}$ (respectively, $\overset{\mathsf{nu.c}}{\approx}$) $U[\{0,1\}^{\ell_{\mathrm{out}}(\lambda)}]$.

3 Type 1: Schemes with Visible Seeds

In this section, we observe (as mentioned in Section 1.1) that the standard security notion for (semi-honest) two-party computation (2PC) is formalized in a way that the internal randomness is visible for adversaries; and consequently, the security of PRGs (where the seed is supposed to be not visible for adversaries) may be unable to in general preserve the security of a protocol when a PRG is applied. We state and prove Theorems 1 and 2 in a more precise manner.

3.1 Basic Definitions

Let π be a 2PC protocol with parties $\mathcal{P}_1$ and $\mathcal{P}_2$ to compute function values $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$ from input pair $\vec{x} = (x_1, x_2)$. Let $\vec{r} = (r_1, r_2)$ be the pair of randomness for $\mathcal{P}_1$ and $\mathcal{P}_2$, $\vec{m}_i(1^\lambda, \vec{x}; \vec{r})$ $(i = 1, 2)$ be the list of messages received by $\mathcal{P}_i$ during the protocol, and $\pi(1^\lambda, \vec{x}; \vec{r})$ denote the pair of outputs by $\mathcal{P}_1$ and $\mathcal{P}_2$ in π. Following the standard formulation (cf. Section 7.2 of [16]), we say that π is *secure against semi-honest* $\mathcal{P}_i$, if there is a PPT simulator $\mathcal{S}_i$ for which $\left(\mathcal{S}_i(1^\lambda, x_i, f_i(\vec{x})), \vec{f}(\vec{x})\right)_{\lambda,\vec{x}} \overset{\mathsf{nu.c}}{\approx} \left(x_i, r_i, \vec{m}_i(1^\lambda, \vec{x}; \vec{r}), \pi(1^\lambda, \vec{x}; \vec{r})\right)_{\lambda,\vec{x}}$ (see Section 2 for the notation $\overset{\mathsf{nu.c}}{\approx}$). We also say "information-theoretically secure", if the relation $\overset{\mathsf{i}}{\approx}$ holds instead of $\overset{\mathsf{nu.c}}{\approx}$.

An important observation is that the internal randomness r_i for party $\mathcal{P}_i$ is included in the input to the distinguisher behind the notation $\overset{\mathsf{nu.c}}{\approx}$. This is practically reasonable, as a corrupted party will be able to see the party's internal randomness for the protocol which is stored in the party's own device.

For a 2PC protocol π, a PRG $\mathcal{R}$, and $i \in \{1, 2\}$, let $\pi \circ_i \mathcal{R}$ denote the modified version of π where, for internal randomness (r'_1, r'_2), party $\mathcal{P}_i$ executes the protocol π with randomness $r_i \leftarrow \mathcal{R}(1^\lambda, r'_i)$, while the other party $\mathcal{P}_{3-i}$ executes π by using randomness $r_{3-i} \leftarrow r'_{3-i}$ as is.

[9] One may think that the seed length of a PRG should satisfy $\ell_{\mathrm{in}}(\lambda) = \lambda$; but our seemingly generalized style is just for the sake of technical ease and our argument can indeed be translated into the more strict style where $\ell_{\mathrm{in}}(\lambda) = \lambda$ always holds.

Algorithm 1: First 2PC protocol π_1 for Theorem 1

```
Input  : ($\mathcal{P}_1$) Blum integer $N$ (as in the text); and randomness $r_1 \in \{0,1\}^{3\lambda}$
         ($\mathcal{P}_2$) $\lambda$-bit prime factors $p < q$ of $N$; and randomness $r_2 \in \{0,1\}^2$
Output: none
1 (By $\mathcal{P}_1$) $y \leftarrow r_1 \bmod N$ and send $y$ to $\mathcal{P}_2$
2 (By $\mathcal{P}_2$) if $y \in \mathsf{QR}_{pq}$ then
3     uniformly sample one of the four square roots $\xi$ of $y \in (\mathbb{Z}/pq\mathbb{Z})^\times$
4     send $\eta \leftarrow \xi$ to $\mathcal{P}_1$
5 else
6     send $\eta \leftarrow \perp$ to $\mathcal{P}_1$
7 end
```

3.2 First Protocol for Theorem 1

We define a 2PC protocol π_1 as in Algorithm 1.[10] For security parameter $\lambda \geq 5$, an input pair is given by $x_1 = N$ and $x_2 = (p, q)$ where $N = pq$ is a Blum integer with λ-bit primes $p < q$ (i.e., $p \equiv q \equiv 3 \pmod 4$). Let $\mathsf{QR}_N = \mathsf{QR}_{pq} \subseteq (\mathbb{Z}/N\mathbb{Z})^\times$ denote the set of quadratic residues modulo $N = pq$. Note that the computation by $\mathcal{P}_2$ is of polynomial time as $\mathcal{P}_2$ has the prime factors p, q of N. Here we focus only on the security against semi-honest $\mathcal{P}_1$, though π_1 is also secure against $\mathcal{P}_2$.

Proposition 1. *π_1 is information-theoretically secure against semi-honest $\mathcal{P}_1$.*

Proof. We consider the PPT simulator $\mathcal{S}$ as in Algorithm 2.[11] We write $\eta = \eta(y)$ in π_1. Moreover, for $y' \in \mathbb{Z}/N\mathbb{Z}$, let $g(y')$ denote the uniform random variable on the set $\{r' \in \{0,1\}^{3\lambda} \mid r' \bmod N = y'\}$ (see also Line 9 of Algorithm 2). Then we have $(r_1, \eta(y)) \overset{\mathsf{i}}{\approx} (g(y), \eta(y))$ by the definition of g. Now, as N is a Blum integer, ± 1 and $\pm a$ in $\mathcal{S}$ are complete representatives for $(\mathbb{Z}/N\mathbb{Z})^\times/\mathsf{QR}_N$. Therefore $y' \overset{\mathsf{i}}{\approx} U[(\mathbb{Z}/N\mathbb{Z})^\times]$ and $\eta^\dagger = \eta(y')$, while $y \overset{\mathsf{i}}{\approx} U[(\mathbb{Z}/N\mathbb{Z})^\times]$ in π_1 as r_1 is λ-bit longer than $N = pq$. Hence $y \overset{\mathsf{i}}{\approx} y'$ and $(g(y), \eta(y)) \overset{\mathsf{i}}{\approx} (g(y'), \eta(y')) \overset{\mathsf{i}}{\approx} (r_1^\dagger, \eta^\dagger)$. Summarizing, we have $(N, r_1, \eta) \overset{\mathsf{i}}{\approx} (N, r_1^\dagger, \eta^\dagger) = \mathcal{S}(1^\lambda, N)$, which implies the claim. □

3.3 First PRG for Theorem 1

We define a PRG for $\mathcal{P}_1$'s randomness in π_1. In order to describe the underlying assumption, first we introduce some terminology. We say that a deterministic

[10] Some reader may feel strange because the two parties' inputs in the protocol are very correlated and the protocol has no output. This is for the sake of simplifying the argument, and in fact our protocol can be converted into a more "natural" but complicated form. See Appendix A for the details.

[11] In fact, in order to let the internal randomness for $\mathcal{S}$ be a *bit sequence*, we have to, and indeed we can, approximate (with exponentially small deviation from the ideal) the procedures in Lines 1, 2, and 9 by PPT algorithms with random bit sequences.

Algorithm 2: Simulator $\mathcal{S}$ for $\mathcal{P}_1$ in protocol π_1

```
   Input  : $1^\lambda$ and $\mathcal{P}_1$'s local input $N$
   Output: $N$, simulated randomness $r_1^\dagger$, and simulated message $\eta^\dagger$ from $\mathcal{P}_2$
1  $x' \leftarrow_R (\mathbb{Z}/N\mathbb{Z})^\times$
2  take some $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ with Jacobi symbol $\left(\frac{a}{N}\right) = -1$
3  $a' \leftarrow_R \{\pm 1, \pm a\}$ and $y' \leftarrow (x')^2 \cdot a' \in (\mathbb{Z}/N\mathbb{Z})^\times$
4  if $y' = (x')^2$ then
5      $\eta^\dagger \leftarrow x'$
6  else
7      $\eta^\dagger \leftarrow \perp$
8  end
9  sample a value $r_1^\dagger \in \{0,1\}^{3\lambda}$ of the uniform random variable, denoted by $g(y')$,
     on the set of all $r' \in \{0,1\}^{3\lambda}$ with $r' \bmod N = y'$
10 return $(N, r_1^\dagger, \eta^\dagger)$
```

polynomial-time algorithm $\mathcal{B} = \mathcal{B}(1^\lambda)$ is a *Blum integer generator*, if its output $\mathcal{B}(1^\lambda)$ (with $\lambda \geq 5$) is a Blum integer with two λ-bit prime factors[12]. We say that $\mathcal{B}$ is *efficiently factorizable*, if there is a PPT *uniform* algorithm $\mathcal{F}$ satisfying that $\mathcal{F}(\mathcal{B}(1^\lambda))$ is a prime factor of $\mathcal{B}(1^\lambda)$ with probability $\Omega(1)$.[13] Then our assumption here is described as follows.

Assumption 1 *There exists a Blum integer generator $\mathcal{B}$ that is not efficiently factorizable; and there exists a non-uniformly secure PRG for any choices of $\ell_{\mathrm{in}}(\lambda)$ and $\ell_{\mathrm{out}}(\lambda)$ (satisfying the constraints in our definition of PRGs)*[14].

Now let $\ell_{\mathcal{S}}(\lambda)$ denote the bit length of the randomness for $\mathcal{S}$. We define $\mathcal{R}_1^*(1^\lambda, r^*)$ for $r^* \in \{0,1\}^{\ell_{\mathcal{S}}(\lambda)}$ to be the second component $r_1^\dagger$ of the output of $\mathcal{S}(1^\lambda, \mathcal{B}(1^\lambda); r^*)$. Then our PRG $\mathcal{R}_1 \colon \{0,1\}^{3\lambda-1} \to \{0,1\}^{3\lambda}$ is defined as follows: first it converts $r_1' \in \{0,1\}^{3\lambda-1}$ to $r^* \in \{0,1\}^{\ell_{\mathcal{S}}(\lambda)}$ by using a PRG $\mathcal{R}_1^\dagger$ as in Assumption 1 (with $\ell_{\mathrm{in}}(\lambda) = 3\lambda - 1$ and $\ell_{\mathrm{out}}(\lambda) = \ell_{\mathcal{S}}(\lambda)$), and then it outputs $\mathcal{R}_1^*(1^\lambda, r^*)$. The PRG satisfies the following:

Proposition 2. *The PRG $\mathcal{R}_1$ is non-uniformly secure.*

Proof. We have $r^* \overset{\mathsf{nu.c}}{\approx} U[\{0,1\}^{\ell_{\mathcal{S}}(\lambda)}]$ by the security of $\mathcal{R}_1^\dagger$, therefore we have $\mathcal{R}_1(1^\lambda, r_1') = \mathcal{R}_1^*(1^\lambda, r^*) \overset{\mathsf{nu.c}}{\approx} \mathcal{R}_1^*(1^\lambda, U[\{0,1\}^{\ell_{\mathcal{S}}(\lambda)}]) \overset{\mathsf{i}}{\approx} U[\{0,1\}^{3\lambda}]$ by Proposition 1 (for $\overset{\mathsf{i}}{\approx}$) and the fact that $\mathcal{R}_1^*$ is PPT (for $\overset{\mathsf{nu.c}}{\approx}$). Hence the claim follows. □

Now we give a precise version of Theorem 1 as follows:

[12] The reason of restricting $\mathcal{B}$ to be deterministic is that $\mathcal{B}$ will be used as a component of the desired PRG and hence may not have its own internal randomness.

[13] The factorization is trivially easy if $\mathcal{F}$ may be non-uniform, as $\mathcal{B}$ is deterministic.

[14] Such a PRG can be obtained from a PRG with 1-bit stretch by a standard technique based on hybrid argument (cf. Construction 3.3.2 and Theorem 3.3.3 of [15]).

Algorithm 3: Non-uniform distinguisher $\mathcal{D}$ for the simulator $\mathcal{S}$ for π_1

```
Input : $1^\lambda$ and $\mathcal{P}_1$'s view $(N, \widehat{r}_1, \widehat{\eta})$, either in real $\pi_1$ or simulated by $\mathcal{S}$
        (also given prime factors $p_\lambda < q_\lambda$ of $N_\lambda \leftarrow \mathcal{B}(1^\lambda)$ as advice)
Output: $b \in \{0,1\}$
1 always return $b \leftarrow 0$ when $N \neq N_\lambda$; below we assume $N = N_\lambda$
2 emulate the protocol $\pi_1$ with inputs $N_\lambda$ and $(p_\lambda, q_\lambda)$ where $\widehat{r}_1$ plays the role of
    randomness for $\mathcal{P}_1$, and get emulated $\mathcal{P}_2$'s message $\overline{\eta}$
3 return $b \leftarrow \chi[\widehat{\eta}, \overline{\eta} \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$ and $(\widehat{\eta})^2 = (\overline{\eta})^2$ and $\overline{\eta} \notin \{\widehat{\eta}, -\widehat{\eta}\}]$
```

Theorem 5. *Under Assumption 1, the protocol π_1 is secure against semi-honest $\mathcal{P}_1$ and the PRG $\mathcal{R}_1$ is non-uniformly secure, but the protocol $\pi_1 \circ_1 \mathcal{R}_1$ is not secure against semi-honest $\mathcal{P}_1$.*

Before giving the proof, we first explain an intuitive idea towards the proof and an outline of the proof. We observe that if $\pi_1 \circ_1 \mathcal{R}_1$ were secure, then for $\mathcal{P}_1$'s input $N = \mathcal{B}(1^\lambda)$ in $\pi_1 \circ_1 \mathcal{R}_1$, $\mathcal{P}_1$ would be unable to obtain any information that cannot be deduced directly from N. In particular, as $\mathcal{B}$ is not efficiently factorizable by Assumption 1, $\mathcal{P}_1$ would be unable to obtain a prime factor of N. However, in fact a corrupted $\mathcal{P}_1$ can factorize N during the protocol $\pi_1 \circ_1 \mathcal{R}_1$ as follows: (1) Given randomness $r'_1 \in \{0,1\}^{3\lambda-1}$, $\mathcal{P}_1$ generates $r^* \in \{0,1\}^{\ell_S(\lambda)}$ as above, and executes $\mathcal{S}(1^\lambda, N; r^*)$ and obtains $(N, r_1^\dagger, \eta^\dagger)$. (2) $\mathcal{P}_1$ executes the protocol π_1 with input N and randomness $r_1^\dagger$, and obtains $\mathcal{P}_2$'s message η (note that this is a correct execution of $\pi_1 \circ_1 \mathcal{R}_1$). (3) If $\eta^\dagger \neq \perp$ and $\eta \neq \pm\eta^\dagger \bmod N$, then $\mathcal{P}_1$ computes $p' \leftarrow \gcd(\eta^2 - (\eta^\dagger)^2, N)$ and outputs p'.

Now if $\eta^\dagger \neq \perp$ (which occurs with probability 1/4), then $\eta^\dagger$ is a square root of $y' = r_1^\dagger \bmod N$. Hence by the construction of π_1, η is one of the four square roots of y', therefore $\eta \neq \pm\eta^\dagger$ occurs with probability 1/2. In this case, we have $\eta^2 - (\eta^\dagger)^2 = (\eta - \eta^\dagger)(\eta + \eta^\dagger)$ and $\eta \pm \eta^\dagger \not\equiv 0 \pmod N$, therefore $\eta - \eta^\dagger$ is divisible by precisely one of the two prime factors of N, which is equal to p'. Hence $\mathcal{P}_1$ can factorize N with probability $\Omega(1)$, a contradiction. This shows the claim.

We start the proof of Theorem 5. Owing to Propositions 1 and 2, it suffices to show that $\pi_1 \circ_1 \mathcal{R}_1$ is not secure against $\mathcal{P}_1$. This follows from the contraposition of the following proposition and Assumption 1 on $\mathcal{B}$.

Proposition 3. *Suppose that the protocol $\pi_1 \circ_1 \mathcal{R}_1$ is secure against $\mathcal{P}_1$. Then there exists a PPT uniform algorithm $\mathcal{F}$ that outputs a prime factor of $\mathcal{B}(1^\lambda)$ with probability $\Omega(1)$.*

Proof. Let $\widetilde{S}$ denote a simulator for $\mathcal{P}_1$ in $\pi_1 \circ_1 \mathcal{R}_1$ implied by the hypothesis. First we consider a PPT non-uniform distinguisher $\mathcal{D}$ in Algorithm 3 for the simulator $\mathcal{S}$ for the protocol π_1, where we let $\chi[P] = 1$ if a condition P holds and $\chi[P] = 0$ otherwise.

When $(N_\lambda, \widehat{r}_1, \widehat{\eta})$ is a view in real π_1, $y \leftarrow \widehat{r}_1 \bmod N_\lambda$ is in QR_N with probability $\approx 1/4$ (where "$\approx$" means "the difference is negligible"). If it is the case, then $\widehat{\eta}$ is a square root of y modulo N_λ. Moreover, in the emulation in Line 2

```
Algorithm 4: Distinguisher $\widetilde{\mathcal{D}}$ for the simulator $\widetilde{\mathcal{S}}$ for $\pi_1 \circ_1 \mathcal{R}_1$
  Input  : $1^\lambda$ and $\mathcal{P}_1$'s view $(N, \widetilde{s}, \widetilde{\eta})$, either in real $\pi_1 \circ_1 \mathcal{R}_1$ or simulated by $\widetilde{\mathcal{S}}$
  Output: $b \in \{0,1\}$
1 always return $b \leftarrow 0$ when $N \neq N_\lambda = \mathcal{B}(1^\lambda)$; below we assume $N = N_\lambda$
2 $(N_\lambda, \widehat{r}_1, \widehat{\eta}) \leftarrow \mathcal{S}(1^\lambda, N_\lambda; \mathcal{R}_1^\dagger(1^\lambda, \widetilde{s}))$
3 return $b \leftarrow \chi[\widehat{\eta}, \widetilde{\eta} \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$ and $(\widehat{\eta})^2 = (\widetilde{\eta})^2$ and $\widetilde{\eta} \notin \{\widehat{\eta}, -\widehat{\eta}\}]$
```

using the *same* randomness $\widehat{r}_1$ for $\mathcal{P}_1$ and *fresh* randomness for $\mathcal{P}_2$, the emulated $\mathcal{P}_1$ sends the same y, while the emulated $\mathcal{P}_2$ replies a uniformly random square root $\overline{\eta}$ of y *independent of* $\widehat{\eta}$. Therefore, when $y \in \mathsf{QR}_N$, we have $b = 1$ with conditional probability $1/2$. Hence $\mathcal{D}$ outputs 1 with probability $\approx 1/8$. Now Proposition 1 implies that $\mathcal{D}$ also outputs 1 with probability $\approx 1/8$ when $(N_\lambda, \widehat{r}_1, \widehat{\eta}) \leftarrow \mathcal{S}(1^\lambda, N_\lambda; s_*)$ with ideally random s_*.

We regard the process "run $\mathcal{D}$ for input $(N_\lambda, \widehat{r}_1, \widehat{\eta}) \leftarrow \mathcal{S}(1^\lambda, N_\lambda; s_*)$" as a PPT non-uniform distinguisher with advice (p_λ, q_λ) against the non-uniformly secure PRG $\mathcal{R}_1^\dagger$. Then it follows that the probability of $b = 1$ is still at least $1/8 - \mathsf{negl}(\lambda) \in \Omega(1)$ when $s_* \leftarrow \mathcal{R}_1^\dagger(1^\lambda, s)$ and s is a uniformly random seed for $\mathcal{R}_1^\dagger$, where negl denotes some negligible function.

For the latter case $(N_\lambda, \widehat{r}_1, \widehat{\eta}) \leftarrow \mathcal{S}(1^\lambda, N_\lambda; s_*)$ with $s_* \leftarrow \mathcal{R}_1^\dagger(1^\lambda, s)$, the component $\widehat{r}_1$ coincides with the output of the PRG $\mathcal{R}_1$ with seed s, therefore the emulated protocol in Line 2 of $\mathcal{D}$ is nothing but the protocol $\pi_1 \circ_1 \mathcal{R}_1$ with randomness s for $\mathcal{P}_1$. Now we consider a PPT distinguisher $\widetilde{\mathcal{D}}$ in Algorithm 4 for simulator $\widetilde{\mathcal{S}}$.

By the argument above, when $(N_\lambda, \widetilde{s}, \widetilde{\eta})$ is a view in real $\pi_1 \circ_1 \mathcal{R}_1$ with input (p_λ, q_λ) for $\mathcal{P}_2$, the probability distribution of $\widetilde{\eta}$ conditioned on the given $(\widetilde{s}, \widehat{r}_1, \widehat{\eta})$ coincides with that of $\overline{\eta}$ in $\mathcal{D}$ for the same $(\widehat{r}_1, \widehat{\eta})$, therefore the probability that $\widetilde{\mathcal{D}}$ outputs $b = 1$ is also $\Omega(1)$ in this case. Now the hypothesis on the simulator $\widetilde{\mathcal{S}}$ implies that the probability of $b = 1$ is also $\Omega(1)$ even when $(N_\lambda, \widetilde{s}, \widetilde{\eta})$ is simulated by $\widetilde{\mathcal{S}}$. That is, by generating $(N_\lambda, \widetilde{s}, \widetilde{\eta}) \leftarrow \widetilde{\mathcal{S}}(1^\lambda, \mathcal{B}(1^\lambda))$ and $(N_\lambda, \widehat{r}_1, \widehat{\eta}) \leftarrow \mathcal{S}(1^\lambda, N_\lambda; \mathcal{R}_1^\dagger(1^\lambda, \widetilde{s}))$, the conditions $\widehat{\eta}, \widetilde{\eta} \in (\mathbb{Z}/N_\lambda\mathbb{Z})^\times$, $(\widehat{\eta})^2 = (\widetilde{\eta})^2$, and $\widetilde{\eta} \notin \{\widehat{\eta}, -\widehat{\eta}\}$ are satisfied with probability $\Omega(1)$; and if it is the case, then a prime factor of N_λ can be found by computing $\gcd(\widetilde{\eta} - \widehat{\eta}, N_\lambda)$. As the aforementioned process of generating $\widetilde{\eta}$ and $\widehat{\eta}$ from $\mathcal{B}(1^\lambda)$ is PPT and uniform, this yields the algorithm $\mathcal{F}$ as in the statement. Hence Proposition 3 holds. □

3.4 Second Protocol and PRG for Theorem 1

We give another pair of a 2PC protocol π_2 and a PRG $\mathcal{R}_2$ for Theorem 1. An outline of the argument is as follows. The protocol π_2 is an oblivious transfer (OT) protocol proposed by Asharov et al. in ACM CCS 2013 [1], or more precisely, Protocol 51 in Section 5.2 of its full version [2]. The key idea of their OT protocol is to construct a function, denoted here by $\mathcal{H}$, that can sample a random element h of an underlying cyclic group $\mathbb{G} = \langle g \rangle$ in a way that the discrete logarithm of h with respect to g is unknown even if the seed used for sampling h is known. Now the Receiver of the 1-out-of-2 OT protocol with input $\sigma \in \{0,1\}$ generates $h \in \mathbb{G}$ by using $\mathcal{H}$ and g^{α} with random α, and sends (g^{α}, h) when $\sigma = 0$ and (h, g^{α}) when $\sigma = 1$ to the Sender. The Sender encrypts the two inputs in a way like the hashed ElGamal encryption where each of the two elements of $\mathbb{G}$ given from the Receiver is used as a public key, and sends the two ciphertexts (c_0, c_1) to the Receiver. Then the Receiver can decrypt c_{σ} and obtain the corresponding input of the Sender as the "secret key" α is known; while the other $c_{1-\sigma}$ cannot be decrypted (hence the other input remains secret) as the "secret key" corresponding to h is not known as mentioned above.

Then our construction of the PRG $\mathcal{R}_2$ is based on the following observation: there is a secure PRG $\mathcal{R}'_2$ that can "cancel" the effect of the function $\mathcal{H}$. Namely, when $h \in \mathbb{G}$ is sampled by $\mathcal{H}$ using an input generated by $\mathcal{R}'_2$ with seed s, now the discrete logarithm of h can be efficiently recovered from s. Then we construct a secure PRG $\mathcal{R}_2$ that involves $\mathcal{R}'_2$ to convert a part s of the seed (s, α) into $\mathcal{R}'_2(s)$. By using the output $(\mathcal{R}'_2(s), \alpha)$ of $\mathcal{R}_2$ in π_2 instead of the Receiver's original randomness, now the Receiver can also decrypt $c_{1-\sigma}$ and break the security, as the corresponding "secret key" can be recovered from s as mentioned above.

Now we move to a precise argument. First, we recall the construction of the OT protocol π_2 mentioned above. To make the argument precise, here we explicitly state that the internal randomness for the two parties are bit sequences, and the uniform samplings of objects in the protocol are performed approximately with exponentially small deviation. The input objects for the protocol (except the security parameter) can be classified into global parameters that can be reused for several protocol executions (such as the underlying cyclic group) and "actual" inputs for each individual protocol execution. For the global parameters, in this paper we put an assumption that a secure global parameter can be chosen efficiently and *deterministically* (see Assumption 2 below). This technical assumption would also have some practical meaning, as it may sometimes happen that an implementation of a protocol hard-wires such a reusable global parameter.

In order to specify our choice of global parameters, we quote the following description from the text in the second paragraph of Section 5.2 in [2] (where "[......]" indicates omission by the author of the present paper):

> [......] *We also assume that it is possible to sample a random element of the group, and the DDH assumption will remain hard even when the coins used to sample the element are given to the distinguisher (i.e., (g, h, g^a, h^a) is indistinguishable from (g, h, g^a, g^b) for random a, b, even given the coins used to sample h).* [......] *For finite fields, one can sample a random element $h \in \mathbb{Z}_p$ of order q by choosing a random $x \in_R \mathbb{Z}_p$ and computing $h = x^{(p-1)/q}$ until $h \neq 1$.* [......]

Accordingly, we use the subgroup of a given order q in the multiplicative group $(\mathbb{F}_p)^{\times}$ of a finite field $\mathbb{F}_p$ (denoted by $\mathbb{Z}_p$ in the quoted text) as the underlying group of the protocol, where p is a t-bit prime for some polynomially bounded $t \geq \lambda$ and q is a divisor of $p-1$. Then the aforementioned sampling method $\mathcal{H}$ for

Algorithm 5: The algorithm $\mathcal{H}$ to sample a subgroup element

```
Input  : $x' \in \{0,1\}^{2t}$
Output: an element $h$ in the order-$q$ subgroup of $(\mathbb{F}_p)^\times$
1 $x \leftarrow x' \bmod p$
2 if $x \neq 0$ then
3     return $h \leftarrow x^{(p-1)/q} \bmod p$
4 else
5     return $h \leftarrow 1$
6 end
```

the group elements can be realized as in Algorithm 5, where slight modification is made in order to ensure that it always halts within finite (and polynomial) time. This algorithm has the following property.

Lemma 1. *The output $\mathcal{H}(x')$ for $x' \leftarrow_R \{0,1\}^{2t}$ is in the unique subgroup of order q in $(\mathbb{F}_p)^\times$ and its probability distribution is exponentially close to uniform over this subgroup.*

Proof. First, if $x = 0$ in the algorithm, then the output h is 1; while if $x \neq 0$, then $h = x^{(p-1)/q} \bmod p$ is an element of $(\mathbb{F}_p)^\times$ of order dividing q, as $(\mathbb{F}_p)^\times$ is a cyclic group of order $p-1$. This implies the former part of the statement. On the other hand, for the latter part of the statement, as the bit length of p is $t \geq \lambda$, the distribution of x is exponentially close to the uniform distribution over $(\mathbb{F}_p)^\times$. Therefore, we may assume without loss of generality that $x \leftarrow_R (\mathbb{F}_p)^\times$. Then $h = x^{(p-1)/q} \bmod p$ becomes a uniformly random element of the subgroup. This implies the latter part of the statement. Hence Lemma 1 holds. □

Our assumption mentioned above, which is a (possibly nonstandard) variant of the decisional Diffie–Hellman (DDH) assumption, is the following:

Assumption 2 *There exists a deterministic polynomial-time algorithm to choose a t-bit prime p with $t \geq \lambda$, a divisor q of $p-1$, a generator g of the subgroup of order q in $(\mathbb{F}_p)^\times$, and a deterministic polynomial-time key derivation function $\mathsf{KDF}\colon \langle g \rangle \to \{0,1\}^L$ for some L, satisfying the following: the two distributions of*

$$(p, q, g, g^r \bmod p, x', \mathsf{KDF}(\mathcal{H}(x')^r \bmod p))$$

and

$$(p, q, g, g^r \bmod p, x', z)$$

with $r \leftarrow_R \{0, \dots, q-1\}$, $x' \leftarrow_R \{0,1\}^{2t}$, $z \leftarrow_R \{0,1\}^L$ are non-uniformly indistinguishable.

Then the protocol π_2 is described in Algorithm 6; here the global parameters are chosen as in Assumption 2 (in particular, the choice of global parameters is deterministic given a security parameter 1^λ). The result in the original paper implies that π_2 is secure in the semi-honest model under Assumption 2.

The following is another precise version of Theorem 1 to be proved here.

Algorithm 6: The OT protocol in [2] (called π_2 here)

```
Input  : (global parameters) $t$-bit prime $p$, divisor $q$ of $p-1$, $g \in (\mathbb{F}_p)^\times$ of
         order $q$, and $\mathsf{KDF}\colon \langle g \rangle \to \{0,1\}^L$
         ($\mathcal{P}_1$ (Sender)) $(x^{\langle 0 \rangle}, x^{\langle 1 \rangle}) \in (\{0,1\}^L)^2$; and randomness $r_1 \in \{0,1\}^{2t}$
         ($\mathcal{P}_2$ (Receiver)) $\sigma \in \{0,1\}$; and randomness $(r'_2, r''_2) \in (\{0,1\}^{2t})^2$
Output: ($\mathcal{P}_1$) none
        ($\mathcal{P}_2$) $\overline{x} \in \{0,1\}^L$                    // To be equal to $x^{\langle \sigma \rangle}$
1  (By $\mathcal{P}_2$) $h \leftarrow \mathcal{H}(r'_2)$ and $\alpha \leftarrow r''_2 \bmod q$
2  (By $\mathcal{P}_2$) if $\sigma = 0$ then
3      $(h^{(0)}, h^{(1)}) \leftarrow (g^\alpha \bmod p, h)$
4  else
5      $(h^{(0)}, h^{(1)}) \leftarrow (h, g^\alpha \bmod p)$
6  end
7  (By $\mathcal{P}_2$) send $(h^{(0)}, h^{(1)})$ to $\mathcal{P}_1$
8  (By $\mathcal{P}_1$) $r \leftarrow r_1 \bmod q$ and $u \leftarrow g^r \bmod p$
9  (By $\mathcal{P}_1$) $(k^{(0)}, k^{(1)}) \leftarrow ((h^{(0)})^r \bmod p, (h^{(1)})^r \bmod p)$
10 (By $\mathcal{P}_1$) $(v^{(0)}, v^{(1)}) \leftarrow \left(x^{(0)} \oplus \mathsf{KDF}(k^{(0)}), x^{(1)} \oplus \mathsf{KDF}(k^{(1)})\right)$
11 (By $\mathcal{P}_1$) send $u$, $v^{(0)}$, and $v^{(1)}$ to $\mathcal{P}_2$
12 (By $\mathcal{P}_2$) return $\overline{x} \leftarrow v^{(\sigma)} \oplus \mathsf{KDF}(u^\alpha \bmod p)$
```

Theorem 6. *Assume that there exists a non-uniformly secure PRG for any choices of $\ell_{\mathrm{in}}(\lambda)$ and $\ell_{\mathrm{out}}(\lambda)$ (satisfying the constraints in PRGs). Assume moreover that the parameters in the protocol π_2 satisfy that $(p-1)/q$ is coprime to q, and that a generator g_0 of $(\mathbb{F}_p)^\times$ can also be chosen in deterministic polynomial time. Then there is a non-uniformly secure PRG $\mathcal{R}_2$ with 1-bit stretch $\ell_{\mathrm{out}}(\lambda) - \ell_{\mathrm{in}}(\lambda) = 1$ satisfying that $\pi_2 \circ_2 \mathcal{R}_2$ is not secure against $\mathcal{P}_2$.*

As mentioned above, the basic strategy for constructing $\mathcal{R}_2$ is to enable $\mathcal{P}_2$ to know the discrete logarithm of $h^{(1-\sigma)} \leftarrow \mathcal{H}(r'_2)$ from the seed for $\mathcal{R}_2$ generating the input r'_2 for $\mathcal{H}$. Then the party $\mathcal{P}_2$ using the PRG $\mathcal{R}_2$ will be able to also unmask $v^{(1-\sigma)}$ by using the seed for $\mathcal{R}_2$ and hence obtain the other $x^{(1-\sigma)}$ as well, violating the security of the OT.

To make the argument precise, we first recall the current assumptions described above: the global parameters p, q, g, and KDF, as well as a generator g_0 of $(\mathbb{F}_p)^\times$, can be *deterministically* chosen in polynomial time, and $(p-1)/q$ is coprime to q. We construct a prototype algorithm $\mathcal{R}_2^*$ for our PRG as in Algorithm 7; our PRG $\mathcal{R}_2$ is then constructed as the composition $\mathcal{R}_2 = \mathcal{R}_2^* \circ \mathcal{R}_2^\dagger \colon \{0,1\}^{4t-1} \to \{0,1\}^{4t}$ where $\mathcal{R}_2^\dagger \colon \{0,1\}^{4t-1} \to \{0,1\}^{9t}$ is a non-uniformly secure PRG implied by the hypothesis of Theorem 6. Now we have the following result on the $\mathcal{R}_2^*$.

Proposition 4. *For $s = (s_1, s_2, s_3, s_4) \leftarrow_R \{0,1\}^{9t}$, the output distribution of $\mathcal{R}_2^*(1^\lambda, s)$ is exponentially close to $U[\{0,1\}^{2t} \times \{0,1\}^{2t}]$, and the e and $r^\dagger$ computed in $\mathcal{R}_2^*$ satisfy that $\mathcal{H}(r^\dagger) = g^e \bmod p$.*

Algorithm 7: The prototype $\mathcal{R}_2^*$ of our PRG

```
  Input  : $1^\lambda$ and seed
           $s = (s_1, s_2, s_3, s_4) \in \{0,1\}^{2t} \times \{0,1\}^{2t} \times \{0,1\}^{3t} \times \{0,1\}^{2t}$
  Output: $(r_2', r_2'') \in \{0,1\}^{2t} \times \{0,1\}^{2t}$
1 choose $p$, $q$, $g$, KDF, and $g_0$ deterministically as in the text
2 compute the multiplicative inverse $d$ of $(p-1)/q$ modulo $q$
3 $e \leftarrow s_1 \bmod q$
4 $h^\dagger \leftarrow g^e \bmod p$
5 $e' \leftarrow s_2 \bmod (p-1)$
6 $h^{\dagger\dagger} \leftarrow (h^\dagger)^d \cdot {g_0}^{qe'} \bmod p$
7 $r^\dagger \leftarrow h^{\dagger\dagger} + (s_3 \bmod K) \cdot p$ where $K = \lfloor (2^{2t} - 1 - h^{\dagger\dagger})/p \rfloor + 1$
  // we have $0 \le r^\dagger \le 2^{2t} - 1$
8 return $(r_2', r_2'') \leftarrow (r^\dagger, s_4)$          // identify $r^\dagger$ with a $2t$-bit sequence
```

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

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

as $h^\dagger \in \langle g \rangle$ and $d \cdot (p-1)/q \equiv 1 \pmod{q}$. Hence we have $\mathcal{H}(r^\dagger) = g^e \bmod p$ by the construction of $\mathcal{H}$, as desired.

For the former part of the statement, it suffices to show that the distribution of $r^\dagger$ is exponentially close to uniform over $\{0,1\}^{2t}$. Let $f \in \{0,\dots,p-2\}$ be the discrete logarithm of g with respect to g_0. Then f is a multiple of $(p-1)/q$ as $g^q = 1$ in $\mathbb{F}_p$; we put $f = f'(p-1)/q$ with $1 \le f' \le q-1$. Now both f' and $(p-1)/q$ are coprime to q, so is f.

As s_1 and s_2 are of $2t$-bit lengths and $t \ge \lambda$, the distributions of e and e' are exponentially close to uniform over $\{0,\dots,q-1\}$ and $\{0,\dots,p-2\}$, respectively. Hence we assume from now that $e \leftarrow_R \{0,\dots,q-1\}$ and $e' \leftarrow_R \{0,\dots,p-2\}$ without loss of generality.

We have $h^{\dagger\dagger} = g^{ed} \cdot {g_0}^{qe'} = {g_0}^{fed+qe'}$ in $\mathbb{F}_p$. Let $\beta = fed + qe' \bmod (p-1)$. Then we have $\beta \bmod q = e \cdot fd \bmod q \in \{0,\dots,q-1\}$. As fd is coprime to q by the argument above, $\beta \bmod q$ is uniformly random (as well as e) and independent of e'. On the other hand, we have $\lfloor \beta/q \rfloor = e' + \lfloor fed/q \rfloor \bmod ((p-1)/q)$. As $e' \leftarrow_R \{0,\dots,p-2\}$, it follows that the pair $(\beta \bmod q, \lfloor \beta/q \rfloor)$ is also uniformly random, so is β. Hence $h^{\dagger\dagger} = {g_0}^\beta$ is uniformly random over $(\mathbb{F}_p)^\times$.

Moreover, as s_3 has $3t$-bit length and $t \ge \lambda$, it follows that, given an $h^{\dagger\dagger}$, the conditional distribution of $r^\dagger$ is exponentially close to uniform over the set $\{r_2' \in \{0,1\}^{2t} \mid r_2' \bmod p = h^{\dagger\dagger}\}$. This implies that, if the distribution of $r_2' \bmod p$ with $r_2' \leftarrow_R \{0,1\}^{2t}$ were identical to the uniform distribution of $h^{\dagger\dagger}$, then the distribution of $r^\dagger$ would be exponentially close to uniform over $\{0,1\}^{2t}$. In fact, as p has t-bit length and $t \ge \lambda$, the distribution of $r_2' \bmod p$ is exponentially close to uniform; therefore the distribution of $r^\dagger$ is indeed exponentially close to uniform, as desired. Hence the former part of the statement holds. This completes the proof of Proposition 4. □

The former part of Proposition 4 and the non-uniform security of $\mathcal{R}_2^{\dagger}$ imply that our PRG $\mathcal{R}_2 = \mathcal{R}_2^{*} \circ \mathcal{R}_2^{\dagger}$ is also non-uniformly secure. Moreover, when party $\mathcal{P}_2$ in the protocol π_2 uses the PRG $\mathcal{R}_2$ with seed $\widetilde{s}$ to generate the internal randomness $(r_2', r_2'') = (r^{\dagger}, s_4) \leftarrow \mathcal{R}_2^{*}(1^{\lambda}, s)$ with $s \leftarrow \mathcal{R}_2^{\dagger}(1^{\lambda}, \widetilde{s})$, the element h is equal to $\mathcal{H}(r_2') = \mathcal{H}(r^{\dagger}) = g^e \bmod p$ and its discrete logarithm e can be recovered from the seed $\widetilde{s}$ for $\mathcal{R}_2$ by computing $s \leftarrow \mathcal{R}_2^{\dagger}(1^{\lambda}, \widetilde{s})$ and then computing e from s as in Line 3 of Algorithm 7. This enables $\mathcal{P}_2$ to obtain $x^{(1-\sigma)}$ as well as $x^{(\sigma)}$ as explained above, which means that now the protocol is not secure against $\mathcal{P}_2$. This completes the proof of Theorem 6.

3.5 Sufficient Conditions for Preserving the Security

To prevent the loss of security as in Theorem 1, here we give some sufficient conditions for a 2PC protocol π and a PRG $\mathcal{R}$ to ensure that $\pi \circ_i \mathcal{R}$ is also secure, as in Theorem 2 in Section 1.1. We introduce the following notion.

Definition 1. *We say that a simulator $\mathcal{S}_i$ for party $\mathcal{P}_i$ is* with raw randomness*, if the randomness for $\mathcal{S}_i$ is of the form (r_i, τ_i) where r_i is the same as the randomness for $\mathcal{P}_i$, and we have $\mathcal{S}_i(1^{\lambda}, x_i, f_i(\vec{x}); r_i, \tau_i) = \langle r_i, \mathcal{T}_{\mathcal{S}_i}(1^{\lambda}, x_i, f_i(\vec{x}), r_i; \tau_i)\rangle$ for a PPT algorithm $\mathcal{T}_{\mathcal{S}_i}$, where the notation $\langle r_i, V_i\rangle$ denotes the simulated view for $\mathcal{P}_i$ consisting of the randomness r_i and the remaining part V_i (here the components in $\langle r_i, V_i\rangle$ are appropriately reordered to keep consistency with the syntax in the definition of a party's view).*

Namely, such a simulator $\mathcal{S}_i$ generates the randomness part of $\mathcal{P}_i$'s view by just outputting a part r_i of $\mathcal{S}_i$'s own randomness, and then $\mathcal{S}_i$ generates the other parts of $\mathcal{P}_i$'s view by using the remaining part τ_i of the randomness (in a way specified by the algorithm $\mathcal{T}_{\mathcal{S}_i}$). For example, the simulator $\mathcal{S}$ in the proof of Proposition 1 for the security of protocol π_1 is *not* with raw randomness (as it generates the randomness part $r_1^{\dagger}$ according to the other part), while the simulator in the security proof of protocol π_2 above given in the original paper [2] is in fact with raw random tape. We give a precise version of Theorem 2.

Theorem 7. *Let π be a 2PC protocol that is information-theoretically secure against a party $\mathcal{P}$ in the semi-honest model where the corresponding simulator is with raw randomness (see above for the terminology). Let $\mathcal{R}$ be a PRG to generate the randomness for $\mathcal{P}$. Suppose moreover that $\ell_{\mathrm{out}}(\lambda) - H_{\infty}(\mathcal{R}(1^{\lambda}, *)) \in O(\log \lambda)$ with uniformly random seed for $\mathcal{R}$ where λ denotes the security parameter and $\ell_{\mathrm{out}}(\lambda)$ denotes the bit length of outputs of $\mathcal{R}$. Then, even by generating the randomness for $\mathcal{P}$ using $\mathcal{R}$, the protocol π remains information-theoretically secure against semi-honest $\mathcal{P}$ and the corresponding simulator is with raw randomness.*

Proof. Let $\mathcal{S}$ be the simulator with raw randomness in the hypothesis. By symmetry, we suppose $\mathcal{P} = \mathcal{P}_1$, and we give a simulator $\widetilde{\mathcal{S}}$ for $\mathcal{P}_1$ in the protocol $\pi \circ_1 \mathcal{R}$ as stated. Put $I = \{0,1\}^{\ell_{\mathrm{in}}(\lambda)}$ and $O = \{0,1\}^{\ell_{\mathrm{out}}(\lambda)}$.

Given 1^{λ}, $\vec{x} = (x_1, x_2)$, a local output o_1 of $\mathcal{P}_1$, and randomness $r_1 \in O$ for $\mathcal{P}_1$ in π, the simulated view for $\mathcal{P}_1$ in π is given by $\langle r_1, \mathcal{T}_{\mathcal{S}}(1^{\lambda}, x_1, o_1, r_1)\rangle$ (see

Definition 1 for the notations). On the other hand, let $V_{\mathrm{real}}(1^\lambda, \vec{x}, r_1)$ denote the random variable of the view for $\mathcal{P}_1$ except the randomness r_1 in a real execution of π with input pair $\vec{x}$ and randomness r_1 for $\mathcal{P}_1$. Then the view for $\mathcal{P}_1$ in a real π is $\langle r_1, V_{\mathrm{real}}(1^\lambda, \vec{x}, r_1) \rangle$. Now we define the simulator $\widetilde{\mathcal{S}}$ in $\pi \circ_1 \mathcal{R}$ as follows:

– Given 1^λ and a local input/output pair (x_1, o_1) as input, $\widetilde{\mathcal{S}}$ chooses $\widetilde{r}_1 \leftarrow_R I$, computes $r_1 \leftarrow \mathcal{R}(1^\lambda, \widetilde{r}_1)$, and outputs $\langle \widetilde{r}_1, \mathcal{T}_{\mathcal{S}}(1^\lambda, x_1, o_1, r_1) \rangle$.

This $\widetilde{\mathcal{S}}$ is with raw randomness by the construction.

Note that the view for $\mathcal{P}_1$ in real $\pi \circ_1 \mathcal{R}$ is given by $\langle \widetilde{r}_1, V_{\mathrm{real}}(1^\lambda, \vec{x}, \mathcal{R}(1^\lambda, \widetilde{r}_1)) \rangle$. Now let Δ and $\widetilde{\Delta}$ denote the statistical distances between the real and simulated views for $\mathcal{P}_1$ in π and in $\pi \circ_1 \mathcal{R}$, respectively, for given 1^λ, $\vec{x} = (x_1, x_2)$, and o_1. Then we have the following (where notations 1^λ are omitted):

$$\begin{aligned}
2\widetilde{\Delta} &= \sum_{\widetilde{s}_1 \in I, V_1} |\Pr[\langle \widetilde{r}_1, \mathcal{T}_{\mathcal{S}}(x_1, o_1, \mathcal{R}(\widetilde{r}_1)) \rangle = \langle \widetilde{s}_1, V_1 \rangle] \\
&\qquad\qquad - \Pr[\langle \widetilde{r}_1, V_{\mathrm{real}}(\vec{x}, \mathcal{R}(\widetilde{r}_1)) \rangle = \langle \widetilde{s}_1, V_1 \rangle]| \\
&= \sum_{\widetilde{s}_1 \in I, V_1} \left| \frac{1}{|I|} \Pr[\mathcal{T}_{\mathcal{S}}(x_1, o_1, \mathcal{R}(\widetilde{s}_1)) = V_1] - \frac{1}{|I|} \Pr[V_{\mathrm{real}}(\vec{x}, \mathcal{R}(\widetilde{s}_1)) = V_1] \right| \\
&= \frac{1}{|I|} \sum_{s_1 \in O, V_1} |I_{s_1}| \cdot |\Pr[\mathcal{T}_{\mathcal{S}}(x_1, o_1, s_1) = V_1] - \Pr[V_{\mathrm{real}}(\vec{x}, s_1) = V_1]|
\end{aligned}$$

where we write $I_{s_1} = \{\widetilde{s}_1 \in I \mid \mathcal{R}(\widetilde{s}_1) = s_1\}$. Now we have $|I_{s_1}|/|I| \le 2^{-H_\infty(\mathcal{R})}$ for each s_1 (where $H_\infty(\mathcal{R}) = H_\infty(\mathcal{R}(1^\lambda, *)))$ by the definition of min-entropy, therefore

$$2\widetilde{\Delta} \le 2^{-H_\infty(\mathcal{R})} \sum_{s_1 \in O, V_1} |\Pr[\mathcal{T}_{\mathcal{S}}(x_1, o_1, s_1) = V_1] - \Pr[V_{\mathrm{real}}(\vec{x}, s_1) = V_1]| \enspace .$$

On the other hand, we have

$$\begin{aligned}
&2\Delta \\
&= \sum_{s_1 \in O, V_1} |\Pr[\langle r_1, \mathcal{T}_{\mathcal{S}}(x_1, o_1, r_1) \rangle = \langle s_1, V_1 \rangle] - \Pr[\langle r_1, V_{\mathrm{real}}(\vec{x}, r_1) \rangle = \langle s_1, V_1 \rangle]| \\
&= \sum_{s_1 \in O, V_1} \left| \frac{1}{|O|} \Pr[\mathcal{T}_{\mathcal{S}}(x_1, o_1, s_1) = V_1] - \frac{1}{|O|} \Pr[V_{\mathrm{real}}(\vec{x}, s_1) = V_1] \right| \\
&= \frac{1}{|O|} \sum_{s_1 \in O, V_1} |\Pr[\mathcal{T}_{\mathcal{S}}(x_1, o_1, s_1) = V_1] - \Pr[V_{\mathrm{real}}(\vec{x}, s_1) = V_1]| \enspace .
\end{aligned}$$

Hence we have $\widetilde{\Delta} \le 2^{-H_\infty(\mathcal{R})} \cdot |O| \cdot \Delta = 2^{\ell_{\mathrm{out}}(\lambda) - H_\infty(\mathcal{R})} \cdot \Delta$. By the hypothesis, Δ is negligible due to the information-theoretic security of π, and $2^{\ell_{\mathrm{out}}(\lambda) - H_\infty(\mathcal{R})} \in 2^{O(\log \lambda)}$ is polynomially bounded in λ. This implies that $\widetilde{\Delta}$ is also negligible, as desired. This completes the proof of Theorem 7. $\square$

4 Type 2: Non-Recognizable "Bad" Randomness

In this and the next sections, we focus on the correctness for PKE schemes[15] with negligible but non-zero decryption error probability, and point out (as mentioned in Section 1.1) that the use of a secure PRG may violate the correctness.

First we introduce some terminology. A PKE scheme $\mathit{\Pi} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three PPT algorithms as follows; $\mathsf{Gen}(1^\lambda)$ outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of a public key pk and a secret key sk; $\mathsf{Enc}_{\mathsf{pk}}(m)$ for a plaintext m outputs a ciphertext c; and $\mathsf{Dec}_{\mathsf{sk}}(c)$ deterministically outputs either a plaintext or a "decryption failure" symbol $\perp$. We say that a key pair $(\mathsf{pk}, \mathsf{sk})$ for a PKE scheme $\mathit{\Pi} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is $\alpha(\lambda)$-*correct*, if

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m] \geq \alpha(\lambda) \text{ for any plaintext } m$$

where the probability is taken for the randomness in Enc. Here "*perfectly correct*" means 1-correct; we also say that $\mathit{\Pi}$ is *perfectly correct*, if all key pairs are perfectly correct. On the other hand, we say that $(\mathsf{pk}, \mathsf{sk})$ is $\beta(\lambda)$-*erroneous*, if

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) \neq m] \geq \beta(\lambda) \text{ for at least one plaintext } m \ .$$

Here we show the following result, which is a precise version of Theorem 3:

Theorem 8. *Assume that there exist a perfectly correct PKE scheme $\mathit{\Pi}^*$ for any (polynomially bounded) choice of plaintext length*[16] *and a (uniformly or non-uniformly) secure PRG $\mathcal{R}^*$ for any choices of $\ell^*_{\mathrm{in}}(\lambda)$ and $\ell^*_{\mathrm{out}}(\lambda)$ (satisfying the constraints in PRGs). Then there exists a pair of a PKE scheme $\mathit{\Pi} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a secure PRG $\mathcal{R}$ with the following two properties:*

- *The original* Gen *generates a not perfectly correct key pair with only exponentially small probability.*
- *When the PRG $\mathcal{R}$ is used in* Gen*, all key pairs generated by the resulting* Gen *are* 1*-erroneous.*

Proof. We assume that $\ell^*_{\mathrm{out}}(\lambda) - \ell^*_{\mathrm{in}}(\lambda) \geq \lambda$ for the PRG $\mathcal{R}^*$ and that the PKE scheme $\mathit{\Pi}^* = (\mathsf{Gen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ has plaintext space $\{0,1\}^{\ell^*_{\mathrm{in}}(\lambda)}$ in the hypothesis of the theorem. We construct the PKE scheme $\mathit{\Pi}$ in the theorem by modifying $\mathit{\Pi}^*$ as follows:

- A public key pk for $\mathit{\Pi}$ consists of a public key pk^* for $\mathit{\Pi}^*$ and $r \leftarrow_R \{0,1\}^{\ell^*_{\mathrm{out}}(\lambda)}$; $\mathsf{pk} = (\mathsf{pk}^*, r)$. The secret key $\mathsf{sk} = \mathsf{sk}^*$ is not changed.
- For a plaintext $m \in \{0,1\}^{\ell^*_{\mathrm{in}}(\lambda)}$, the encryption algorithm Enc first checks if $\mathcal{R}^*(1^\lambda, m) = r$ or not. If $\mathcal{R}^*(1^\lambda, m) \neq r$, then encryption and decryption are performed in the same way as $\mathit{\Pi}^*$. If $\mathcal{R}^*(1^\lambda, m) = r$, then Enc outputs a broken ciphertext (say $\perp$) which always yields decryption error.

[15] As the security is not the central topic here, we just implicitly assume IND-CPA security for the PKE schemes in the following arguments.

[16] Again, such a PKE scheme can be obtained via a hybrid argument (cf. Section 5.2.5.3 of [16]) from a perfectly correct PKE scheme with 1-bit plaintexts.

As $\ell^*_{\mathrm{out}}(\lambda) - \ell^*_{\mathrm{in}}(\lambda) \geq \lambda$, the probability that the component r of pk is in the range of $\mathcal{R}^*$ is at most $2^{-\lambda}$. As the behavior of Π coincides with Π^* whenever r is not in the range of $\mathcal{R}^*$, the requirement for correctness of Π is satisfied.

We define the PRG $\mathcal{R}$ in a way that, it ideally samples the internal randomness r_{gen} for Gen^* and samples $r \in \{0,1\}^{\ell^*_{\mathrm{out}}(\lambda)}$ by $r \leftarrow \mathcal{R}^*(1^\lambda, s)$ with $s \leftarrow_R \{0,1\}^{\ell^*_{\mathrm{in}}(\lambda)}$; $(r_{\mathsf{gen}}, r) \leftarrow \mathcal{R}(1^\lambda, (r_{\mathsf{gen}}, s))$.[17] Then the modified key generation algorithm chooses the components pk^* and r of pk by using the two output components of $\mathcal{R}$, respectively. Note that the security of $\mathcal{R}^*$ implies the security of $\mathcal{R}$ straightforwardly. Now for any public key $\mathsf{pk} = (\mathsf{pk}^*, r)$ in Π generated by using $\mathcal{R}$ with seed (r_{gen}, s) as above, we have $r = \mathcal{R}^*(1^\lambda, s)$ by the construction, therefore decryption error will occur with probability 1 for plaintext $m = s$. Hence, now any key pair for Π is 1-erroneous, and the claim holds. $\square$

5 Type 3: Implicit Non-Uniform Distinguishers

In this section, we continue to focus on the correctness for PKE schemes with negligible errors, but here we deal with the randomness in the encryption algorithm instead of the key generation studied in the previous section. We point out the implicit relation to non-uniform security of PRGs, and show the following result which is a precise version of Theorem 4:

Theorem 9. *Assume that there exist a perfectly correct PKE scheme Π^* for any (polynomially bounded) choice of plaintext length. Assume moreover that there exists a uniformly secure PRG $\mathcal{R}^*$ that is not non-uniformly secure, for any choices of $\ell^*_{\mathrm{in}}(\lambda)$ and $\ell^*_{\mathrm{out}}(\lambda)$ (satisfying the constraints in PRGs). Then there exist a PKE scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a uniformly secure PRG $\mathcal{R}$ with the following two properties:*

- *All key pairs of Π are $(1 - \varepsilon(\lambda))$-correct for an exponentially small $\varepsilon(\lambda)$.*
- *When the PRG $\mathcal{R}$ is used in Enc of Π, all key pairs are $\beta(\lambda)$-erroneous with respect to the resulting Enc for a non-negligible $\beta(\lambda)$.*

We explain an outline of the proof. First, by the hypothesis on $\mathcal{R}^*$, there is a PPT non-uniform distinguisher $\mathcal{D}^*$ for $\mathcal{R}^*$ with non-negligible advantage. We assume that the PKE scheme $\Pi^* = (\mathsf{Gen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ in the hypothesis has plaintext space involving the advice for $\mathcal{D}^*$. The PKE scheme Π has the same key generation and decryption algorithms as Π^*.

The encryption algorithm Enc for Π is defined by modifying Enc^* as follows. For the internal randomness, two blocks called Block k ($k = 0, 1$) of polynomially many random bit sequences is added, each of which follows a probability distribution X_k. Originally, X_0 and X_1 are identical and uniform. Then, given a plaintext m, Enc first tries to distinguish the distributions X_0 and X_1 by using the polynomially many random samples provided in Blocks 0 and 1. Here Enc uses the distinguisher $\mathcal{D}^*$ with advice m. If $\mathcal{D}^*$ detects a significant bias between

[17] The technical constraint for $\mathcal{R}$ that the seed length should be a strictly increasing function of λ can be ensured by adjusting the seed length of $\mathcal{R}^*$.

the two blocks then Enc outputs a broken ciphertext (say $\perp$) that always yields decryption error; otherwise Enc encrypts m in the same way as Enc^*.

In the original Enc, X_0 and X_1 are identical, therefore (if the size of two blocks is sufficiently large) $\mathcal{D}^*$ detects a significant bias with only exponentially small probability whatever the plaintext (the advice for $\mathcal{D}^*$) is. This implies the first condition in the statement. On the other hand, we construct the PRG $\mathcal{R}$ in a way that $\mathcal{R}$ replaces the distribution X_0 with the output distribution of $\mathcal{R}^*$ while it keeps the distribution X_1 unchanged (the standard hybrid argument implies that $\mathcal{R}$ is uniformly secure as well as $\mathcal{R}^*$). When the $\mathcal{R}$ is applied to Enc (denoted by Enc'), $\mathcal{D}^*$ with the correct advice m can distinguish the output distribution X_0 of $\mathcal{R}^*$ from the uniform distribution X_1, therefore (if the size of two blocks is sufficiently large) the $\mathcal{D}^*$ inside Enc' detects a significant bias with non-negligible probability. As this case yields decryption error, the decryption error probability of Enc' for the plaintext m becomes non-negligible, implying the second condition in the statement. Hence the claim holds.

Now we move to a precise proof of the theorem.

Proof (Theorem 9). First, by the hypothesis on $\mathcal{R}^*$, there is a PPT non-uniform distinguisher $\mathcal{D}^*$ for $\mathcal{R}^*$ with non-negligible advantage; that is, there are an integer $k \geq 1$ and infinitely many λ's for which the advantage is larger than λ^{-k}. We focus on those λ's from now on. Let $Q(\lambda)$ be a polynomial bound for the length of advice which the PPT $\mathcal{D}^*$ can read. We assume that the PKE scheme $\Pi^* = (\mathsf{Gen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ in the hypothesis has plaintext space $\{0,1\}^{Q(\lambda)}$. The PKE scheme Π has the same key generation and decryption algorithms as Π^*.

The encryption algorithm Enc for Π is defined as in Algorithm 8, where we set $\rho(\lambda) = 16\lambda^{2k+1}$ and $\theta(\lambda) = 8\lambda^{k+1}$. Roughly summarizing, the internal randomness for Enc involves (besides the other components) uniformly random $\ell^*_{\mathrm{out}}(\lambda)$-bit sequences $r_{i,j}$ with $i \in \{0,1\}$ and $1 \leq j \leq \rho(\lambda)$. Before encrypting plaintext m, for each i, Enc runs $\mathcal{D}^*$ (with randomly fixed prefix m^* of m as advice) $\rho(\lambda)$ times independently for inputs $r_{i,1}, \dots, r_{i,\rho(\lambda)}$ and counts the number μ_i of output bits being 1. If the numbers μ_0 and μ_1 differ at most $\theta(\lambda)$, then Enc encrypts m in the same way as Π^*. Otherwise, Enc outputs a broken ciphertext (say $\perp$) that always yields decryption error.

Intuitively, when the $r_{i,j}$'s are ideally random, all the corresponding output distributions of $\mathcal{D}^*$ are identical, therefore the difference of the numbers of 1's in "$i = 0$ part" and "$i = 1$ part" will be small with high probability, implying the required correctness for Π. Precisely, the opposite condition $|\mu_0 - \mu_1| > \theta(\lambda)$ implies that $|\mu_i - \rho(\lambda) \cdot p_1| > \theta(\lambda)/2$ for at least one $i \in \{0,1\}$, where p_1 denotes the probability that $\mathcal{D}^*$ outputs 1 for a uniformly random input from $\{0,1\}^{\ell^*_{\mathrm{out}}(\lambda)}$. By Hoeffding's Inequality (Lemma 2 below) with $n = \rho(\lambda) = 16\lambda^{2k+1}$ and $nt = \theta(\lambda)/2 = 4\lambda^{k+1}$ (hence $nt^2 = (nt)^2/n = \lambda$), the latter condition holds with probability at most $2 \cdot 2\exp(-2nt^2) = 4e^{-2\lambda}$. Hence the behavior of Π deviates from the correct Π^* with exponentially small probability, as desired.

Lemma 2 (Hoeffding's Inequality [19]). *Let $X_1, \dots, X_n$ be independent random variables, each taking the value* 1 *with probability p and the value* 0 *with*

Algorithm 8: Encryption algorithm Enc for our PKE scheme Π

```
Input  : $1^\lambda$ and plaintext $m \in \{0,1\}^{Q(\lambda)}$
         (the internal randomness involves components $r_{i,j} \in \{0,1\}^{\ell^*_{\mathrm{out}}(\lambda)}$
         with $i \in \{0,1\}$ and $1 \leq j \leq \rho(\lambda)$, as well as the other components)
Output: (possibly broken) ciphertext $c$
1  choose a prefix $m^*$ of $m$ uniformly at random
2  for $i \leftarrow 0$ to $1$ do
3      $\mu_i \leftarrow 0$
4      for $j \leftarrow 1$ to $\rho(\lambda)$ do
5          if $\mathcal{D}^{*(m^*)}(1^\lambda, r_{i,j})$ (with fresh randomness) outputs $1$ then
6              $\mu_i \leftarrow \mu_i + 1$
7          end
8      end
9  end
10 if $|\mu_0 - \mu_1| \leq \theta(\lambda)$ then
11     return $c \leftarrow \mathsf{Enc}^*(m)$
12 else
13     return a broken ciphertext $c$ (yielding decryption error)
14 end
```

probability $1 - p$ *for a common* p*. Then for any* $t > 0$*, we have*

$$\Pr\left[\left|\frac{X_1 + \cdots + X_n}{n} - p\right| \geq t\right] \leq 2\exp\left(-2nt^2\right) \quad .$$

On the other hand, the seed for our PRG $\mathcal{R}$ is the same as the internal randomness for Enc except that the components $r_{0,1}, \dots, r_{0,\rho(\lambda)}$ are replaced with independent and uniformly random $s_1, \dots, s_{\rho(\lambda)} \in \{0,1\}^{\ell^*_{\mathrm{in}}(\lambda)}$. When $\mathcal{R}$ generates the internal randomness for Enc, each $r_{0,j}$ is chosen by $r_{0,j} \leftarrow \mathcal{R}^*(1^\lambda, s_j)$, while the other components, including the $r_{1,j}$'s, are ideally sampled. By a standard hybrid argument, the uniform security of $\mathcal{R}^*$ implies the uniform security of $\mathcal{R}$. (We note that, the technical constraint for the seed length to be a strictly increasing function of λ can be ensured by adding some dummy components to the seed.) Intuitively, as $\mathcal{D}^*$ can distinguish the PRG $\mathcal{R}^*$ from ideal randomness, now the difference of the numbers of 1's in the pseudorandom "$i = 0$ part" and the ideally random "$i = 1$ part" will be large with high probability, which yields a broken ciphertext with high probability as well.

To make the argument precise, let m^* be a prefix of some plaintext m that is the correct advice for $\mathcal{D}^*$ to distinguish $\mathcal{R}^*$. Let p_0 denotes the probability that $\mathcal{D}^*$ outputs 0 for an input $\mathcal{R}^*(1^\lambda, s)$ with $s \leftarrow_R \{0,1\}^{\ell^*_{\mathrm{in}}(\lambda)}$, while p_1 is the same as above. Then the hypothesis on $\mathcal{D}^*$ implies that $|p_0 - p_1| > \lambda^{-k}$ and hence $|\rho(\lambda) \cdot p_0 - \rho(\lambda) \cdot p_1| > \rho(\lambda)\lambda^{-k} = 2\theta(\lambda)$ for this choice of m^*. Now the opposite condition $|\mu_0 - \mu_1| \leq \theta(\lambda)$ implies that $|\mu_i - \rho(\lambda) \cdot p_i| > \theta(\lambda)/2$ for at least one $i \in \{0,1\}$. Hoeffding's Inequality with the same parameters n, t as above also implies that the latter condition holds with probability at most $4e^{-2\lambda}$. By taking into account the choice of m^* among the $Q(\lambda)+1$ candidates, it follows that decryption error occurs for the m with probability at least

$$\beta(\lambda) = \frac{1-4e^{-2\lambda}}{Q(\lambda)+1} \enspace .$$

We moreover set $\beta(\lambda) = 0$ for the remaining λ's not focused in the argument above; the resulting $\beta(\lambda)$ is still a non-negligible function. Hence all key pairs are $\beta(\lambda)$-erroneous when the PRG $\mathcal{R}$ is applied, as desired. This completes the proof of Theorem 9. $\square$

From now, given an individual correct PKE scheme $\mathit{\Pi} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we provide a possible strategy to generically convert (depending on the $\mathit{\Pi}$) a uniformly secure PRG $\mathcal{R}$ into a uniformly secure PRG $\overline{\mathcal{R}}$ that preserves the correctness when applied to generate the randomness for Enc.

We introduce some notations. Let $(\mathsf{pk}, \mathsf{sk})$ be a key pair for $\mathit{\Pi}$ with security parameter λ, let m be a plaintext, and let $r \in \{0,1\}^{L(\lambda)}$ where $L(\lambda)$ is the length of randomness for Enc. We define a function $\mathcal{F}_{\lambda,\mathsf{pk},\mathsf{sk},m,r} \colon \{0,1\}^{L(\lambda)} \to \{0,1\}$ by

$$\mathcal{F}_{\lambda,\mathsf{pk},\mathsf{sk},m,r}(r^\dagger) = \begin{cases} 0 & \text{if } \mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; r \oplus r^\dagger)) = m \enspace , \\ 1 & \text{if } \mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; r \oplus r^\dagger)) \neq m \enspace . \end{cases}$$

We say that a PRG $\mathcal{R}^\dagger$ with output length $\ell_{\mathrm{out}}^\dagger(\lambda) = L(\lambda)$ *$\eta(\lambda)$-fools the function family $\mathcal{F}$*, if for any $\mathsf{ind} = (\lambda, \mathsf{pk}, \mathsf{sk}, m, r)$ as above, we have

$$\left| \Pr\left[\mathcal{F}_{\mathsf{ind}}(\mathcal{R}^\dagger(1^\lambda, U[\{0,1\}^{\ell_{\mathrm{in}}^\dagger(\lambda)}])) = 1 \right] - \Pr\left[\mathcal{F}_{\mathsf{ind}}(U[\{0,1\}^{L(\lambda)}]) = 1 \right] \right| \leq \eta(\lambda) \enspace .$$

Then we define the PRG[18] $\overline{\mathcal{R}}$ with seed $\overline{s} = (s, s^\dagger) \in \{0,1\}^{\ell_{\mathrm{in}}(\lambda)} \times \{0,1\}^{\ell_{\mathrm{in}}^\dagger(\lambda)}$ by

$$\overline{\mathcal{R}}(1^\lambda, \overline{s}) = \mathcal{R}(1^\lambda, s) \oplus \mathcal{R}^\dagger(1^\lambda, s^\dagger) \enspace .$$

Such an XOR-ing construction of a PRG combining two PRGs of different types has been studied in the literature in some different contexts; for example, this is similar to the "dual-mode PRG" in [25]. Now if $\mathcal{R}^\dagger$ is PPT, then the security $\mathcal{R}(1^\lambda, U[\{0,1\}^{\ell_{\mathrm{in}}(\lambda)}]) \overset{\mathsf{u.c}}{\approx} U[\{0,1\}^{L(\lambda)}]$ of $\mathcal{R}$ implies that

$$\overline{\mathcal{R}}(1^\lambda, U[\{0,1\}^{\overline{\ell}_{\mathrm{in}}(\lambda)}]) \overset{\mathsf{u.c}}{\approx} U[\{0,1\}^{L(\lambda)}] \oplus \mathcal{R}^\dagger(1^\lambda, U[\{0,1\}^{\ell_{\mathrm{in}}^\dagger(\lambda)}]) = U[\{0,1\}^{L(\lambda)}] \enspace ,$$

i.e., $\overline{R}$ is uniformly secure. Moreover, we have the following result.

Theorem 10. *Suppose that the PRG $\mathcal{R}^\dagger$ $\eta(\lambda)$-fools the function family $\mathcal{F}$ (see above for the terminology) and a key pair $(\mathsf{pk}, \mathsf{sk})$ of $\mathit{\Pi}$ with security parameter λ is $\alpha(\lambda)$-correct. Then, when the randomness for Enc is generated by the PRG $\overline{\mathcal{R}}$, the key pair $(\mathsf{pk}, \mathsf{sk})$ becomes $(\alpha(\lambda) - \eta(\lambda))$-correct.*

[18] We assume that the PRG $\overline{\mathcal{R}}$ satisfies the constraint $\overline{\ell}_{\mathrm{in}}(\lambda) = \ell_{\mathrm{in}}(\lambda) + \ell_{\mathrm{in}}^\dagger(\lambda) < \overline{\ell}_{\mathrm{out}}(\lambda) = L(\lambda)$ for input/output lengths.

Proof. Let m be any plaintext. We have to evaluate the probability

$$\begin{aligned}\varepsilon &= \Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; \overline{\mathcal{R}}(1^\lambda, U[\{0,1\}^{\overline{\ell}_{\mathrm{in}}(\lambda)}]))) \neq m] \\ &= \sum_s 2^{-\ell_{\mathrm{in}}(\lambda)} \Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; \mathcal{R}(1^\lambda, s) \oplus \mathcal{R}^\dagger(1^\lambda, U[\{0,1\}^{\ell^\dagger_{\mathrm{in}}(\lambda)}]))) \neq m] \\ &= \sum_s 2^{-\ell_{\mathrm{in}}(\lambda)} \Pr[\mathcal{F}_{\mathcal{R}(1^\lambda,s)}(\mathcal{R}^\dagger(1^\lambda, U[\{0,1\}^{\ell^\dagger_{\mathrm{in}}(\lambda)}])) = 1]\end{aligned}$$

where s runs over $\{0,1\}^{\ell_{\mathrm{in}}(\lambda)}$ and we write $\mathcal{F}_r = \mathcal{F}_{\lambda,\mathsf{pk},\mathsf{sk},m,r}$. Now, as $\mathcal{R}^\dagger$ $\eta(\lambda)$-fools the function family $\mathcal{F}$ by the hypothesis, we have

$$\begin{aligned}\varepsilon &\leq 2^{-\ell_{\mathrm{in}}(\lambda)} \sum_s \Big(\Pr[\mathcal{F}_{\mathcal{R}(1^\lambda,s)}(U[\{0,1\}^{L(\lambda)}]) = 1] + \eta(\lambda) \Big) \\ &= \eta(\lambda) + 2^{-\ell_{\mathrm{in}}(\lambda)} \sum_s \Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; \mathcal{R}(1^\lambda, s) \oplus U[\{0,1\}^{L(\lambda)}])) \neq m] \enspace .\end{aligned}$$

As each $\mathcal{R}(1^\lambda, s) \oplus U[\{0,1\}^{L(\lambda)}]$ is identical to $U[\{0,1\}^{L(\lambda)}]$, it follows that

$$\varepsilon \leq \eta(\lambda) + \Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) \neq m] \leq \eta(\lambda) + (1 - \alpha(\lambda)) = 1 - (\alpha(\lambda) - \eta(\lambda))$$

by the hypothesis on $(\mathsf{pk}, \mathsf{sk})$. This implies the claim. □

Theorem 10 reduces our task to develop a “special-purpose” PRG $\mathcal{R}^\dagger$ that fools the *explicitly restricted* function family $\mathcal{F}$. The complexity of each function in the family is almost the sum of complexity of Enc, Dec, and the given PRG $\mathcal{R}$, which will be fairly small when the PKE scheme Π and the PRG $\mathcal{R}$ are efficient. Developing a PRG fooling this function family might be a relatively easier task than developing a non-uniformly secure PRG, the latter having to fool *any* non-uniform distinguisher with *arbitrarily large* (polynomially bounded) *complexity*. To develop such a special-purpose PRG $\mathcal{R}^\dagger$, some techniques in the area of derandomization such as those in [24, 27] would be useful.

A A “Natural” Variant of Algorithm 1

Here we give a “natural” variant of 2PC protocol π_1 defined in Section 3.2 (Algorithm 1) where the inputs for two parties are not correlated and the parties have outputs in the protocol. The modified protocol is given in Algorithm 9. Here $\mathcal{F}_{\mathsf{EQ}}$ denotes an ideal functionality for two-party equality test, where the common output $\beta = 1$ (respectively, $\beta = 0$) means that the two inputs are equal (respectively, not equal).

In the part of the protocol before executing π_1, the two parties check if their inputs satisfy the required conditions in the original protocol π_1. More precisely, first, the input (p, q) for $\mathcal{P}_2$ in π_1 should satisfy that $p < q$, p and q are primes, and $p \equiv q \equiv 3 \pmod 4$. In the protocol here, $\mathcal{P}_2$ first checks if these conditions hold, and if it fails then the protocol halts at this step. Secondly, assuming the

Algorithm 9: A variant of protocol π_1

```
  Input  : ($\mathcal{P}_1$) a non-negative integer $N < 2^{2\lambda}$
           ($\mathcal{P}_2$) $\lambda$-bit integers $p, q$
  Output: (common to $\mathcal{P}_1$ and $\mathcal{P}_2$) an integer $\iota \in \{0, 1, 2\}$
1 (By $\mathcal{P}_2$) if $p \geq q$, or $p$ or $q$ is not a prime $\equiv 3 \pmod 4$ then
2   | halt the protocol, where both parties output $\iota = 2$
3 end
4 (By $\mathcal{P}_1$ and $\mathcal{P}_2$) execute $\mathcal{F}_{\mathsf{EQ}}(N, pq)$ and obtain common output $\beta$
5 if $\beta = 0$ then
6   | halt the protocol, where both parties output $\iota = 1$
7 end
8 (By $\mathcal{P}_1$ and $\mathcal{P}_2$) execute the protocol $\pi_1$ with inputs $N$ and $(p, q)$
9 halt the protocol, where both parties output $\iota = 0$
```

Algorithm 10: Implementation of the functionality $\mathcal{F}_{\mathsf{EQ}}$

```
  Input  : ($\mathcal{P}_i$ $(i = 1, 2)$) a non-negative integer $x_i < 2^{2\lambda}$
  Output: (common to $\mathcal{P}_1$ and $\mathcal{P}_2$) a bit $\beta$
1 (By $\mathcal{P}_1$) generate a key pair $(\mathsf{pk}, \mathsf{sk})$ for the lifted-ElGamal cryptosystem
   $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathbb{F}_P$ of prime order $P > 2^{2\lambda+1}$
2 (By $\mathcal{P}_1$) send $\mathsf{pk}$ and $\mathsf{Enc}(-x_1)$ to $\mathcal{P}_2$
3 (By $\mathcal{P}_2$) generate $r \boxdot (\mathsf{Enc}(x_2) \boxplus \mathsf{Enc}(-x_1)) \boxplus \mathsf{Enc}(0) = \mathsf{Enc}(r(x_2 - x_1))$ for
   $r \leftarrow_R (\mathbb{F}_P)^{\times}$, and send $c \leftarrow \mathsf{Enc}(r(x_2 - x_1))$ to $\mathcal{P}_1$
4 (By $\mathcal{P}_1$) if $c$ is a ciphertext of plaintext $0$ then
5   | halt the protocol, where both parties output $\beta = 1$
6 else
7   | halt the protocol, where both parties output $\beta = 0$
8 end
```

conditions for $\mathcal{P}_2$'s input, the input N for $\mathcal{P}_1$ should satisfy that $N = pq$. This condition is checked by using $\mathcal{F}_{\mathsf{EQ}}$, and if it fails then the protocol halts at this step. Once these conditions have been verified, the parties can execute the protocol π_1 with the correct input pair. By focusing on the input pairs satisfying the conditions in π_1, the protocol here inherits from π_1 the property that the security will be lost by applying a certain secure PRG to the randomness for $\mathcal{P}_1$.

For the security against $\mathcal{P}_1$, if the output is $\iota = 2$, then $\mathcal{P}_1$ receives no message and hence the security holds trivially. If $\iota = 1$, then $\mathcal{P}_1$ just participates in the execution of $\mathcal{F}_{\mathsf{EQ}}$ and obtains the output $\beta = 0$, therefore the security follows from the security of $\mathcal{F}_{\mathsf{EQ}}$. Finally, if $\iota = 0$, then $\mathcal{P}_1$ participates in the execution of $\mathcal{F}_{\mathsf{EQ}}$ with output being always $\beta = 1$ and also participates in π_1, therefore the security also follows from the security of $\mathcal{F}_{\mathsf{EQ}}$ and π_1.

For the sake of completeness, we describe in Algorithm 10 a well-known implementation of $\mathcal{F}_{\mathsf{EQ}}$ using the lifted-ElGamal cryptosystem. Here $\boxplus$ and $\boxdot$ denote the homomorphic addition and homomorphic scalar multiplication, respectively. We analyze the behavior of the protocol as follows:

– If $x_1 = x_2$, then the ciphertext c in the protocol is a random ciphertext of plaintext $r(x_2-r_1) = 0$ (note that the randomness in c has also been perfectly rerandomized, as a random ciphertext $\mathsf{Enc}(0)$ was homomorphically added). Hence the protocol outputs the correct value $\beta = 1$, and now the message received by $\mathcal{P}_1$ is a random ciphertext $\mathsf{Enc}(0)$ as mentioned above, which can be perfectly simulated.
– If $x_1 \neq x_2$, then $x_2 - x_1 \in (\mathbb{F}_P)^\times$ by the property $P > 2^{2\lambda+1}$, while $r \leftarrow_R (\mathbb{F}_P)^\times$. Therefore the plaintext $r(x_2 - x_1)$ for c is also uniformly random over $(\mathbb{F}_P)^\times$ and hence the protocol outputs the correct value $\beta = 0$ (note that, though $r(x_2 - x_1)$ can be large and the lifted-ElGamal cryptosystem enables to efficiently decrypt small plaintexts only, the protocol just checks if the ciphertext c has plaintext 0 or not, which is still efficiently checkable). Moreover, in this case, the received message c is a random ciphertext for a uniformly random non-zero plaintext, which can be perfectly simulated.

Hence the correctness and the security (against $\mathcal{P}_1$) of the implemented $\mathcal{F}_{\mathsf{EQ}}$ have been verified. In particular, the security against $\mathcal{P}_1$ is information-theoretic. Therefore, as well as the original protocol π_1, the variant of π_1 given here also has information-theoretic security against $\mathcal{P}_1$, as desired.

Acknowledgements. The author thanks all the members of a study group "Shin-Akarui-Angou-Benkyoukai", in particular Shota Yamada, Tadanori Teruya, Kazumasa Shinagawa, Takashi Yamakawa, Takahiro Matsuda, Yusuke Sakai, Keita Emura, and Goichiro Hanaoka, for fruitful discussions. This work was supported by JST PRESTO Grant Number JPMJPR14E8, by JST CREST Grant Number JPMJCR14D6, and by JST CREST Grant Number JPMJCR19F6.

References

1. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 535–548, 2013.
2. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. *IACR Cryptology ePrint Archive*, 2013:552, 2013.
3. Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 1–20, 2011.
4. Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, pages 321–340, 2013.

5. Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, pages 592–606, 2017.
6. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.
7. Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 649–665, 2010.
8. Jean Paul Degabriele, Kenneth G. Paterson, Jacob C. N. Schuldt, and Joanne Woodage. Backdoors in pseudorandom number generators: Possibility and impossibility results. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 403–432, 2016.
9. Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 101–126, 2015.
10. Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 196–205, 2004.
11. Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud, and Daniel Wichs. Security analysis of pseudo-random number generators with input: /dev/random is not robust. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 647–658, 2013.
12. Yevgeniy Dodis and Yanqing Yao. Privacy with imperfect randomness. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 463–482, 2015.
13. Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 342–360, 2004.
14. Oded Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *J. Cryptol.*, 6(1):21–53, 1993.
15. Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.
16. Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
17. Carmit Hazay and Hila Zarosim. The feasibility of outsourced database search in the plain model. In *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*, pages 313–332, 2016.
18. Shoichi Hirose. Secure block ciphers are not sufficient for one-way hash functions in the preneel-govaerts-vandewalle model. In *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, pages 339–352, 2002.

19. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
20. Thomas Holenstein and Renato Renner. One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 478–493, 2005.
21. Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 226–246, 2003.
22. Pavel Hub'avcek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 163–172, 2015.
23. Huijia Lin and Stefano Tessaro. Amplification of chosen-ciphertext security. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 503–519, 2013.
24. Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.
25. Koji Nuida. How to use pseudorandom generators in unconditional security settings. In *Provable Security - 8th International Conference, ProvSec 2014, Hong Kong, China, October 9-10, 2014. Proceedings*, pages 291–299, 2014.
26. Krzysztof Pietrzak and Maciej Skorski. Non-uniform attacks against pseudoentropy. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 39:1–39:13, 2017.
27. Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 648–657, 2001.