Electronic Colloquium on Computational Complexity, Report No. 187 (2015)

A Note on Perfect Correctness by Derandomization[*]

Nir Bitansky Vinod Vaikuntanathan

Abstract

In this note, we show how to transform a large class of erroneous cryptographic schemes into perfectly correct ones. The transformation works for schemes that are correct on every input with probability noticeably larger than half, and are secure under parallel repetition. We assume the existence of one-way functions and of functions with deterministic (uniform) time complexity $2^{O(n)}$ and non-deterministic circuit complexity $2^{\Omega(n)}$. The transformation complements previous results showing that public-key encryption and indistinguishability obfuscation that err on a noticeable fraction of inputs can be turned into ones that are often correct *for all inputs*.

The technique relies on the idea of "reverse randomization" [Naor, Crypto 1989] and on Nisan-Wigderson style derandomization, which was previously used in cryptography to obtain non-interactive witness-indistinguishable proofs and commitment schemes [Barak, Ong and Vadhan, Crypto 2003].

[*]MIT. E-mail: {nirbitan,vinodv}@csail.mit.edu. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, the NEC Corporation, and a Steven and Renee Finn Career Development Chair from MIT.

ISSN 1433-8092

1 Introduction

Randomized algorithms are, by their very nature, error-prone, yet for some problems they are significantly faster (and often simpler) than their state-of-the-art deterministic counterparts. This gap has motivated a rich study of *derandomization*, where a central avenue has been the design of *pseudo-random generators* [BM84, Yao82a, NW94] that could offer one universal solution for the problem. This has led to surprising results, intertwining cryptography and complexity theory, and culminating in a derandomization of **BPP** under worst-case complexity assumptions (the existence of functions in $\mathbf{E} = \mathbf{Dtime}(2^{O(n)})$ with *worst-case* circuit complexity $2^{\Omega(n)}$) [NW94, IW97].

For cryptographic algorithms the picture is somewhat more subtle. Indeed, in cryptography, randomness is almost always needed in order to guarantee any sense of security. In particular, cryptographic schemes are often *perfectly correct* even if randomized. Yet, in some cases cryptographic algorithm do make errors. For example, in some encryption algorithms, notably the lattice-based ones [AD97, Reg05], most but not all ciphertexts can be decrypted correctly. Here, however, we cannot resort to general derandomization, as a (completely) derandomized version will most likely be totally insecure.

While for general algorithms infrequent errors are tolerable in practice, for cryptographic algorithms errors can be (and have been) exploited by adversaries (see [BDL01] and a long line of followup works). Thus, the question of eliminating errors is ever more important in the cryptographic context. This question was addressed in a handful of special contexts in cryptography. In the context of interactive proofs, [GMS87, FGM$^+$89] show how to turn any interactive proof into one with perfect completeness. In the context of encryption schemes, Goldreich Goldwasser and Halevi [GGH97] showed how to *partially* eliminate errors from lattice-based encryption schemes [AD97, Reg05], and Dwork, Naor and Reingold [DNR04a] show how to *partially* eliminate errors from *any* encryption scheme. Here, "partial" refers to the fact that they eliminate errors from the encryption and decryption algorithms, but not the key generation algorithm. That is, in their final *immunized* encryption scheme, it could still be the case that there are bad keys that always cause decryption errors.

This work. We show how to *completely immunize* a large class of cryptographic algorithms into ones that make no errors at all. Our most general result concerns cryptographic algorithms (or protocols) that are secure under parallel repetition. We show:

Theorem 1.1 (Informal)**.** *Assume the existence of one-way functions and of functions with deterministic (uniform) time complexity $2^{O(n)}$ and non-deterministic circuit complexity $2^{\Omega(n)}$. Then, any encryption scheme, indistinguishability obfuscation scheme, and (parallel repetition secure) multiparty computation protocol can be completely immunized against errors.*

Our tools, perhaps unsurprisingly, come from the area of derandomization, in particular we make heavy use of Nisan-Wigderson (NW) type pseudorandom generators. Such NW-generators were previously used by Barak, Ong and Vadhan [BOV07] to remove interaction from commitment schemes and ZAPs. We use it here for a different purpose of immunizing cryptographic algorithms from errors to make them perfectly correct. Below, we elaborate on the similarities and differences.

1.1 The Basic Idea

We briefly explain the basic idea behind the transformation, focusing on the case of public-key encryption. Imagine that we have an encryption scheme given by randomized key-generation and encryption algorithms, and a deterministic decryption algorithm $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where for any message $m \in \{0,1\}^n$ there is a tiny decryption error:

$$\Pr_{(r_g,r_e)\leftarrow\{0,1\}^{\mathrm{poly}(n)}} \left[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m;r_e)) \neq m \mid (pk,sk) = \mathsf{Gen}(r_g)\right] \leq 2^{-n} \enspace .$$

Can we deterministically choose "good randomness" (r_g, r_e) that leads to correct decryption? This question indeed seems analogous to the question of derandomizing **BPP**. There, the problem can be solved using Nisan-Wigderson type pseudo-random generators [NW94]. Such generators can produce a poly(n)-long pseudo-random string using a short random seed of length $d(n) = O(\log n)$. They are designed to fool distinguishers of some prescribed polynomial size $t(n)$, and may run in time $2^{O(d)} \gg t$. Derandomization of the **BPP** algorithm is then simply done by enumerating over all $2^d = n^{O(1)}$ seeds and taking the majority.

We can try to use NW-type generators to solve our problem in a similar way. However, the resulting scheme wouldn't be secure – indeed, it will be *deterministic*, which means it cannot be semantically secure [GM84]. To get around this, we use the idea of reverse randomization from [Lau83, Nao91, DN07, DNR04b]. For each possible seed $i \in \{0,1\}^d$ for the NW-generator NWPRG, we derive corresponding randomness

$$(r_e^i, r_g^i) = \mathsf{NWPRG}(i) \oplus (\mathsf{BMYPRG}(s_e^i), \mathsf{BMYPRG}(s_g^i)) \enspace .$$

Here BMYPRG is a Blum-Micali-Yao (a.k.a cryptographic) pseudo-random generator [BM82, Yao82b], and the seeds $(s_g^i, s_e^i) \in \{0,1\}^{n^{\Omega(1)}}$ are chosen independently for every i, with the sole restriction that their image is sparse enough (say, has size at most $2^{n/2}$). Encryption and decryption for any given message are now done in parallel with respect to all 2^d copies of the original scheme, where the final result of decryption is defined to be the majority of the 2^d decrypted messages.

Security is now guaranteed by the BMY-type generators and the fact that public-key encryption can be securely performed in parallel. Crucially, the pseudo-randomness of BMY strings is guaranteed despite the fact that their image forms a sparse set; in particular, when shifted at random this set will completely evade the (tiny) set of "bad randomness" (that lead to decryption errors). In the above construction, the image is not shifted truly at random, but rather by an NW-pseudo-random string, and we would like to argue that this suffices to get the desired correctness.

To argue that NW-pseudo-randomness is enough, we need to show that with high-probability over the choice of the NW string, the shifted image of the BMY generator still evades "bad randomness". This may not be efficiently testable deterministically, but can be tested non-deterministically, by guessing the seeds for the BMY generator that would lead to bad randomness. We thus rely on NW generators that fool non-deterministic circuits. Such pseudo-random generators are known under the worst case assumption that there exist functions in **E** with non-deterministic circuit complexity $2^{\Omega(n)}$ [SU01].

Relation to [BOV07]. Barak, Ong, and Vadhan were the first to demonstrate how NW-type derandomization can be useful in cryptography. They showed how NW generators can be used to derandomize Naor's commitments [Nao91] and Dwork and Naor's ZAPs [DN07]. In the applications they examined, "reverse randomization" is already encapsulated in the constructions of ZAPs and commitments that they start from, and they show that “the random shift” can be derandomized, using the fact that ZAPs and commitments are secure under parallel repetition.

There, they were not interested in the correctness of a specific computation *per se*, but rather in the *existence* of an “incorrect object”, namely an accepting proof for a false statement in ZAPs, or a commitment with inconsistent openings. (The above intuition captures this more general type of correctness.) Another difference is that in the applications they consider, it is in fact enough to use hitting set generators (against co-non-determinism) rather than pseudorandom generators. Intuitively, the reason is that in these applications there is one-sided error. For example, in a ZAP system, one already assumes that true statements are always accepted by the verifier, so when derandomizing they only need to recognize false statements. This is analogous to having an encryption system that is always correct on encryptions of zero, but may make mistakes on encryptions of one.

Organization. Below, we formulate the above transformation for a general class of cryptographic schemes in protocols. Section 2 gives a minimal model for schemes and protocols that we will use to describe the transformation. Section 3 presents the transformation itself. In Section 4, we briefly discuss several examples of interest where the transformation can be applied.

2 Cryptographic Schemes and Protocols

We consider a simple model of cryptographic schemes and protocols that will allow to describe the transformation generally. In Section 4, we give several examples of schemes and protocols and how they fit into the framework.

Honest Executions: Let λ be a security parameter and let $m = m(\lambda), n = n(\lambda), \ell = \ell(\lambda)$. An honest execution of an m-party scheme (or protocol) Π involves interaction between m PPT parties with inputs $(x_1, \dots, x_m) \in \{0,1\}^{n \times m}$ and randomness $(r_1, \dots, r_m) \in \{0,1\}^{\ell \times m}$, at the end of which they each produce outputs $(y_1, \dots, y_m) \in \{0,1\}^{n \times m}$. Abstracting out, we will think of Π as a single process that runs in some fixed polynomial time and denote it by $y \leftarrow \Pi(1^\lambda, x, r)$, where $x = (x_1, \dots, x_m), y = (y_1, \dots, y_m)$, and $r = (r_1, \dots, r_m)$.

Definition 2.1 ($(1-\alpha)$-correctness)**.** *Let $f : \{0,1\}^{n \times m} \to \{0,1\}^{n \times m}$ be a function computable by circuits of size* $\mathrm{poly}(\lambda)$*. Π computes f $(1-\alpha)$-correctly if for any λ and any $x \in \{0,1\}^{n \times m}$,*

$$\Pr_{r \leftarrow \{0,1\}^{\ell \times m}} \left[y \neq f(x) \;\middle|\; y \leftarrow \Pi(1^\lambda, x, r) \right] \leq \alpha(\lambda) \ .$$

Repeated Executions: For a function $k = k(\lambda)$, inputs $x = (x_1, \dots, x_m) \in \{0,1\}^{n \times m}$ and randomness $r = (r_{ij})_{i \in [m], j \in [k]}$, and $r_{i,j} \in \{0,1\}^\ell$, the repeated execution $\Pi_{\otimes k}(1^\lambda, x, r)$ consists of executing $\Pi(1^\lambda, x, r_1), \dots, \Pi(1^\lambda, x, r_k)$, where $r_j = (r_{1j}, \dots, r_{mj})$, in parallel and obtaining corresponding outputs $y = (y_{ij})_{i \in [m], j \in [k]}$.

Executions in Presence of Adversaries: We consider a simple model of executions in the presence of an adversary. Such an execution is modeled as a non-uniform family Π^* of $\mathrm{poly}(\lambda)$ size and denoted by $y^* \leftarrow \Pi^*(1^\lambda, r)$, where y^* represents some arbitrary adversarial view, and r represents randomness for honest parties.

3 Removing Errors

In this section, we define the basic tools required for the main transformation and present the transformation itself.

3.1 NW and BMY PRGs

We define NW-type PRGs [NW94] and BMY-type PRGs [BM82, Yao82b].

Definition 3.1 (Nondeterministic Circuits)**.** *A nondeterministic boolean circuit $C(x,w)$ takes x as a primary input and w as a witness. We define $C(x) := 1$ if and only if there exists w such that $C(x,w) = 1$.*

Definition 3.2 (NW-Type PRGs against Nondeterministic Circuits)**.** $\mathsf{NWPRG} : \{0,1\}^{d(n)} \to \{0,1\}^n$ *is an NW-generator against non-deterministic circuits of size $t(n)$ if it is computable in time $2^{O(d(n))}$ and any non-deterministic circuit C of size at most $t(n)$ distinguishes $U \leftarrow \{0,1\}^n$ from* $\mathsf{NWPRG}(n)$*, where $s \leftarrow \{0,1\}^{d(n)}$, with advantage at most $1/t(n)$.*

We shall rely on the following theorem by Shaltiel and Umans regarding the existence NW-type PRGs as above assuming worst-case hardness for non-deterministic circuits.

Theorem 3.3 ([SU01])**.** *Assume there exists a function in $\mathbf{E} = \mathbf{Dtime}(2^{O(n)})$ with nondeterministic circuit complexity $2^{\Omega(n)}$. Then, for any polynomial $t(\cdot)$, there exists an NW-generator* NWPRG : $\{0,1\}^{d(n)} \to \{0,1\}^n$ *against non-deterministic circuits of size $t(n)$, where $d(n) = O(\log n)$.*

We remark that the above worst-case assumption can be seen as a natural generalization of the assumption that $\mathbf{EXP} \not\subseteq \mathbf{NP}$. We also note that there is a universal candidate for the corresponding PRG, by instantiating the the hard function with any $\mathbf{E}$-complete language under linear reductions. See further discussion in [BOV07].

We now define BMY-type (a.k.a cryptographic) PRGs.

Definition 3.4 (BMY-Type PRGs)**.** *A function* $\mathsf{BMYPRG} : \{0,1\}^{d(n)} \to \{0,1\}^n$ *is a BMY-generator if it is computable in time* $\mathrm{poly}(d(n))$ *and any $n^{O(1)}$-size non-uniform family distinguishes $U \leftarrow \{0,1\}^n$ from* $\mathsf{BMYPRG}(n)$*, where $s \leftarrow \{0,1\}^{d(n)}$, with advantage at most $n^{-\omega(1)}$.*

Theorem 3.5. *[HILL99] BMY-type pseudo-random generators can be constructed from any one-way function.*

3.2 The Transformation

We now describe a transformation from any $(1-\alpha)$-correct scheme Π for a function f into a perfectly correct one. For a simpler exposition, we restrict attention to the case that the error α is tiny. We later explain how this restriction can be removed.

Ingredients. In the following, let λ be a security parameter, let $m = m(\lambda), n = n(\lambda), \ell = \ell(\lambda)$ be polynomials, and $\alpha = \alpha(\lambda) \leq 2^{-(\lambda+n)m-2}$. We rely on the following:

- A $(1-\alpha)$-correct scheme Π computing $f : \{0,1\}^{n\times m} \to \{0,1\}^{n\times m}$ where each party uses randomness of length ℓ.

- A BMY-type pseudo-random generator $\mathsf{BMYPRG} : \{0,1\}^\lambda \to \{0,1\}^\ell$.
- An NW-type pseudo-random generator $\mathsf{NWPRG} : \{0,1\}^d \to \{0,1\}^{\ell \times m}$ against nondeterministic circuits of size $t = t(\lambda)$, where t and d depend on $m, n, \ell, \Pi, f, \mathsf{BMYPRG}$, $8 \leq t = \lambda^{O(1)}$, and $d(\lambda) = O(\log \lambda)$, and will be specified later on. We shall denote $k = 2^d$.

The New Scheme:

Given security parameter 1^λ and input $x \in \{0,1\}^{n \times m}$:

1. **Randomness Generation:** Each party $i \in [m]$
   - samples k BMY strings $(r_{i1}^{\mathsf{BMY}}, \ldots, r_{ik}^{\mathsf{BMY}})$, where $r_{ij}^{\mathsf{BMY}} = \mathsf{BMYPRG}(s_{ij})$ and $s_{ij} \leftarrow \{0,1\}^\lambda$.
   - computes (all) k NW strings $(r_1^{\mathsf{NW}}, \ldots, r_k^{\mathsf{NW}})$, where $r_j^{\mathsf{NW}} = \mathsf{NWPRG}(j)$, and derives $(r_{i1}^{\mathsf{NW}}, \ldots, r_{ik}^{\mathsf{NW}})$, where r_{ij} is the ith ℓ-bit block of r_j^{NW}.
   - compute $r_{i1}, \ldots, r_{ik}$ where $r_{ij} = r_{ij}^{\mathsf{BMY}} \oplus r_{ij}^{\mathsf{NW}}$.
2. **Emulating the Parallel Scheme:**
   - the parties emulate the repeated scheme $\Pi_{\otimes k}(1^\lambda, x, r)$, with randomness $r = (r_{ij})_{i \in [m], j \in [k]}$.
   - each party i obtains outputs $(y_{i1}, \ldots, y_{ik})$ and outputs $y_i = \mathsf{majority}(y_{i1}, \ldots, y_{ik})$.

Correctness. We now turn to show that the new scheme is perfectly correct.

Proposition 3.1. *The new scheme is perfectly-correct.*

Proof. We first claim that, if we sample r^{NW} truly at random from $\{0,1\}^{\ell \times m}$, then except with probability $1/4$, there exist no $s_1, \ldots, s_m \in \{0,1\}^\lambda$ and $x \in \{0,1\}^{n \times m}$ such that $f(x) \neq \Pi(1^\lambda, x, r)$ for $r = r^{\mathsf{NW}} \oplus r^{\mathsf{BMY}}$ and $r^{\mathsf{BMY}} = (\mathsf{BMYPRG}(s_1), \ldots, \mathsf{BMYPRG}(s_m))$. Indeed, in this case for any fixed $s_1, \ldots, s_m$ and fixed x, the string r is truly random and the scheme is guaranteed to error with probability at most $\alpha \leq 2^{-(\lambda+n)m-2}$. The claim thus follows by taking a union bound over all $2^{m\lambda}$ tuples $s_1, \ldots, s_m$ and 2^{nm} inputs x.

We now claim that the same holds, except with probability $\frac{1}{4} + \frac{1}{t} \leq \frac{3}{8}$, when $r^{\mathsf{NW}} = \mathsf{NWPRG}(j)$ for $j \leftarrow \{0,1\}^d$, namely r^{NW} is pseudo-random and not truly random. This will conclude the proof since it means that the majority of the $k = 2^d$ parallel executions of Π will always be correct.

To prove this claim, let us assume towards contradiction that it does not hold. We construct a non-deterministic distinguisher that breaks NWPRG. The distinguisher, given r^{NW}, non-deterministically guesses $s_1, \ldots, s_m$ and x, computes $r^{\mathsf{BMY}} = (\mathsf{BMYPRG}(s_1), \ldots, \mathsf{BMYPRG}(s_m))$, $r = r^{\mathsf{NW}} \oplus r^{\mathsf{BMY}}$, and checks whether $f(x) \neq \Pi(1^\lambda, x, r)$. Recall that when r^{NW} is truly random such a witness $s_1, \ldots, s_m, x$ exists with probability at most $1/4$, whereas, by our assumption towards contradiction, when r^{NW} is pseudo-random such a witness exists with probability at least $\frac{1}{t} + \frac{1}{4}$.

The size of the above distinguisher is some fixed polynomial $t'(\lambda)$ that depends only on m, n, ℓ and the time required to compute Π, f, BMYPRG. Thus in the construction we choose $t > t'$ (and d is chosen accordingly), meaning that the constructed distinguisher indeed breaks NWPRG. $\square$

Security. We now observe that the new scheme is as secure as the k-fold repeated scheme when using true randomness. Examples are given in the next section.

Concretely, we consider two distributions on randomness $r^b = (r_{ij}^b)_{i\in[m],j\in[k],b\in\{0,1\}}$ for the parties in $\Pi_{\otimes k}$:

1. r_{ij}^0 is sampled truly at random (and independently) from $\{0,1\}^\ell$.

2. r_{ij}^1 is computed as in the above scheme; namely $r_{ij} = r_{ij}^{\mathsf{BMY}} \oplus r_{ij}^{\mathsf{NW}}$, where $r_{i,j}^{\mathsf{BMY}} = \mathsf{BMYPRG}(s_{ij})$ for $s_{ij} \leftarrow \{0,1\}^\lambda$ and r_{ij}^{NW} is a fixed string (we don't care how it's computed for this part).

Proposition 3.2. *For any non-uniform adversarial execution $\Pi_{\otimes k}^*$ of size $\lambda^{O(1)}$*

$$\Pi_{\otimes k}^*(1^\lambda, r^0) \approx_c \Pi_{\otimes k}^*(1^\lambda, r^1) \ .$$

Proof. Follows directly from the security of the BMY PRG and a standard hybrid argument. □

Removing the Assumption Regarding Tiny Error. Above we assumed that $\alpha(\lambda) \leq 2^{-(\lambda+n)m-2}$. We can start from any $\alpha \leq \frac{1}{2} - \eta$, for $\eta = \lambda^{-O(1)}$, perform $k' = O((\lambda+n)m\eta^{-2})$ repetitions to reduce the error, and then apply the above transformation.

The amount of randomness $\ell(\lambda)$, and the execution time, grow proportionally, but are still polynomial in λ. Also, the same security guarantee as above holds, except that we should consider the $k \times k'$-fold repetition of Π, rather than the k fold one. This is sufficient as long as the original scheme was secure (in some sense) for *any* polynomial number of repetitions.

4 Examples of Interest

We briefly mention three examples of interest and how they fit into the above framework.

Public-Key Encryption. Public-key encryption can be modeled as a three-party scheme consisting of a generator, an encryptor, and a decryptor. The generator has no input, and uses its randomness r_1 to generate pk and sk, which are sent to the encryptor and decryptor, respectively. The encryptor has as input a message m, and uses its randomness r_2 in order to generate an encryption $\mathsf{Enc}_{pk}(m; r_1)$, which is sent to the decryptor. The decryptor has no input nor randomness, it uses the secret key to decrypt and outputs the decrypted message. (In this case the function computed by Π is $f(\perp, m, \perp) = (\perp, \perp, m)$.)

In the repeated scheme $\Pi_{\otimes k}$, k independent encryptions $\mathsf{Enc}_{pk}(m; r_{11}^0), \ldots, \mathsf{Enc}_{pk}(m; r_{1k}^0)$ are generated. For any two messages $m, m' \in \{0,1\}^n$, we can consider adversarial executions $\Pi_{\otimes k}^m(1^\lambda, r^0)$ and $\Pi_{\otimes k}^{m'}(1^\lambda, r^0)$, which output as the adversarial view the corresponding encryptions. By the semantic security of the encryption scheme (plus a standard hybrid argument) and Proposition 3.2, we deduce that the corrected scheme (using r^1 generated as in the transformation) is also semantically secure:

$$\Pi_{\otimes k}^m(1^\lambda, r^1) \approx_c \Pi_{\otimes k}^m(1^\lambda, r^0) \approx_c \Pi_{\otimes k}^{m'}(1^\lambda, r^0) \approx_c \Pi_{\otimes k}^{m'}(1^\lambda, r^1) \ .$$

We note that in [DN07], Dwork, Naor, and Reingold show how public-key encryption where decryption errors may even occur for a large fraction of messages, can be transformed into ones that only have a tiny decryption error over the randomness of the scheme. Applying our transformation, we can further turn such schemes into perfectly correct ones.

Indistinguishability Obfuscation. Indistinguishability obfuscation [BGI$^+$12] can be modeled as a two-party scheme consisting of an obfuscator and an evaluator. The obfuscator has as input a circuit C, and uses its randomness r_1 in order to create an obfuscation $\mathcal{O}(C)$, which is sent to the evaluator. The evaluator has an input x for the circuit, and no randomness, it computes $\mathcal{O}(C)(x)$ and outputs the result. (In this case the function computed by Π is $f(C, x) = (\perp, C(x))$.)

In the repeated scheme $\Pi_{\otimes k}$, k independent obfuscations $\mathcal{O}(C; r^0_{11}), \ldots, \mathcal{O}(C; r^0_{1k})$ are generated. For any two circuits $C, C' \in \{0,1\}^n$ that compute the same function, we can consider adversarial executions $\Pi^C_{\otimes k}(1^\lambda, r^0)$ and $\Pi^{C'}_{\otimes k}(1^\lambda, r^0)$, which output as the adversarial view the corresponding obfuscations. As in the case of encryption, by the security of the obfuscation scheme (plus a standard hybrid argument) and Proposition 3.2, we deduce that the corrected scheme (using r^1 generated as in the transformation) is also secure.

We note that in [BV16], Bitansky and Vaikuntanathan show how indistinguishability obfuscation [BGI$^+$12] where the obfuscated circuit may error on a large fraction of inputs, can be transformed into one that only has a tiny error over the randomness of the obfuscator. Applying our transformation, we can further turn such schemes into perfectly correct ones. We now briefly explain how this is captured in the above framework.

MPC. The framework also naturally capture an MPC protocol for computing an m-party f. If the repeated protocol $\Pi_{\otimes k}$ is secure against static attacks so is the corrected protocol. Here we'll consider an adversarial execution $\Pi^{\mathcal{A}}_{\otimes k}$ that outputs the view of $\mathcal{A}$ and the outputs of honest parties. If $\mathcal{A}$ has a simulator when the honest parties use randomness generated as r^0, then the same simulator works for the corrected scheme (where r^1 is used).

References

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 284–293. ACM, 1997.

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001.

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 112–117, 1982.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[BOV07] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[BV16] Nir Bitansky and Vinod Vaikuntanthan. Indistinguishability obfuscation: from approximate to exact. In *Theory of Cryptography - 13th Theory of Cryptography Conference, TCC 2016, Tel Aviv, Israel, January 10-13, 2016*, 2016.

[CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[DNR04a] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Cachin and Camenisch [CC04], pages 342–360.

[DNR04b] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Cachin and Camenisch [CC04], pages 342–360.

[FGM+89] Martin Furer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. 5:429–442, 1989.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Eliminating decryption errors in the ajtai-dwork cryptosystem. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 105–111. Springer, 1997.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMS87] Oded Goldreich, Yishay Mansour, and Michael Sipser. Interactive proof systems: Provers that never fail and random selection (extended abstract). In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 449–461. IEEE Computer Society, 1987.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229, 1997.

[Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17(4):215–217, 1983.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[SU01] 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.

[Yao82a] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982.

[Yao82b] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.