Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 187 (2015)

A Note on Perfect Correctness by Derandomization*

Nir Bitansky Vinod Vaikuntanathan

Abstract

We show a general compiler that transforms a large class of erroneous cryptographic schemes (such as public-key encryption, indistinguishability obfuscation, and secure multiparty computation schemes) into perfectly correct ones. The transformation works for schemes that are *correct on all inputs 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 $2^{\Omega(n)}$.

Our transformation complements previous results that showed how public-key encryption and indistinguishability obfuscation that err on a noticeable fraction of inputs can be turned into ones that *for all inputs* are often correct.

The technique relies on the idea of "reverse randomization" [Naor, Crypto 1989] and on Nisan-Wigderson style derandomization, previously used in cryptography to remove interaction from 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 often faster and simpler than their state-of-the-art deterministic counterparts, yet, by their very nature, they are error-prone. 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, namely, 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 *necessary* to guarantee any sense of security. While many cryptographic schemes are *perfectly correct* even if randomized, some 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.

It gets worse. 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]. Subsequent works, starting from that of Dwork, Naor and Reingold [DNR04a], show how to *partially* eliminate errors from *any* encryption scheme [HR05, LT13]. 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. In the context of indistinguishability obfuscation (IO), Bitansky and Vaikuntanathan [BV16] recently showed how to partially eliminate errors from any IO scheme: namely, they show how to convert any IO scheme that might err on a fraction of the inputs into one that is correct on all inputs, with high probability over the coins of the obfuscator.

This Work. We show how to *completely immunize* a large class of cryptographic algorithms, turning them into algorithms 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 that one-way functions exist and functions with deterministic (uniform) time complexity $2^{O(n)}$ and non-deterministic circuit complexity $2^{\Omega(n)}$ exist. Then, any encryption scheme, indistinguishability obfuscation scheme, and multiparty computation protocol that is secure under parallel repetition can be completely immunized against errors.*

More precisely, we show that perfect correctness is guaranteed when the transformed scheme or protocol are executed honestly. The security of the transformed scheme or protocol is inherited from the security of the original scheme under parallel repetition. In the default setting of encryption and obfuscation schemes, encryption and obfuscation are always done honestly, and security under parallel repetition is well known to be guaranteed automatically. Accordingly, we obtain the natural notion of perfectly-correct encryption and obfuscation. In contrast, in the setting of MPC,

corrupted parties may in general affect any part of the computation. In particular, in the case of corrupted parties, the transformed protocol does not provide a better correctness guarantee, but only the same correctness guarantee as the original (repeated) protocol. We find that perfect correctness is a natural requirement and the ability to generically achieve it for a large class of cryptographic schemes is aesthetically appealing. In addition, while in many applications almost perfect correctness may be sufficient, some applications do require *perfectly correct* cryptographic schemes. For example, using public-key encryption as a commitment scheme requires perfect correctness, the construction of non-interactive witness-indistinguishable proofs in [BP15] requires a perfectly correct indistinguishability obfuscation, and the construction of 3-message zero knowledge against uniform verifiers [BCPR14], requires perfectly correct delegation schemes.

Our tools, perhaps unsurprisingly given the above discussion, 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, namely to immunize cryptographic algorithms from errors. 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:

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

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 $\mathrm{poly}(n)$-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

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

Here BMYPRG is a Blum-Micali-Yao (a.k.a cryptographic) pseudo-random generator [BM82, Yao82b], and the seeds $(s_g^i, s_e^i) \in \{0,1\}^\ell$ are chosen independently for every i, with the sole restriction that their image is sparse enough (say, they are of total length $\ell = n/2$). 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. The fact that the set of BMY string is sparse will be used to the perfect correctness of the scheme. In particular, when shifted at random, this set will evade the (tiny) set of "bad randomness" (that lead to decryption errors) with high probability $1 - 2^{\ell-n} \geq 1 - 2^{-n/2}$.

In the actual 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 enough probability (say 0.51) over the choice of the NW string, the shifted image of the BMY generator still evades "bad randomness". This last property may not be efficiently testable deterministically, but can be tested non-deterministically in fixed polynomial time, by guessing the seeds for the BMY generator that would lead to bad randomness. We accordingly 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 $\mathbf{E}$ 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. 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. In Section 2, we give the required preliminaries. Section 3 presents the transformation itself. In Section 4, we discuss several examples of interest where the transformation can be applied.

2 Preliminaries

In this section, we give the required preliminaries, including standard computational concepts, cryptographic schemes and protocols, and the derandomization tools that we use.

2.1 Standard Computational Concepts

We recall standard computational concepts concerning Turing machines and Boolean circuits.

- By *algorithm* we mean a uniform Turing machine. We say that an algorithm is PPT if it is probabilistic and polynomial time.

- A polynomial-size circuit family $\mathcal{C}$ is a sequence of circuits $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$, such that each circuit C_λ is of polynomial size $\lambda^{O(1)}$ and has $\lambda^{O(1)}$ input and output bits.
- We follow the standard habit of modeling any efficient adversary strategy $\mathcal{A}$ as a family of polynomial-size circuits. For an adversary $\mathcal{A}$ corresponding to a family of polynomial-size circuits $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, we often omit the subscript λ, when it is clear from the context. For simplicity, we shall simply call such an adversary a *polynomial-size adversary*.
- We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if it decays asymptotically faster than any polynomial.
- Two ensembles of random variables $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are said to be computationally indistinguishable, denoted by $\mathcal{X} \approx_c \mathcal{Y}$, if for all polynomial-size distinguishers $\mathcal{D}$, there exists a negligible function ν such that for all λ,

$$|\Pr[\mathcal{D}(X_\lambda) = 1] - \Pr[\mathcal{D}(Y_\lambda) = 1]| \leq \nu(\lambda).$$

2.2 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 such schemes and protocols.

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

Definition 2.1 ($(1-\alpha)$-Correctness)**.** *Let $f : \{0,1\}^{n\times m} \to \{0,1\}^{n\times m}$ be a polynomial-time computable function. 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 $x = (x_1, \ldots, x_m) \in \{0,1\}^{n\times m}$ and randomness $r = (r_{ij})_{i\in[m],j\in[k]}$, and $r_{i,j} \in \{0,1\}^\ell$, the repeated execution $y \leftarrow \Pi_{\otimes k}(1^\lambda, x, r)$ consists of executing $\Pi(1^\lambda, x, r_1), \ldots, \Pi(1^\lambda, x, r_k)$, where $r_j = (r_{1j}, \ldots, r_{mj})$, in parallel and obtaining the corresponding outputs, namely, $y = (y_{ij})_{i\in[m],j\in[k]}$.

2.3 NW and BMY PRGs

We now define the basic tools required for the main transformation — NW-type PRGs [NW94] and BMY-type PRGs [BM82, Yao82b]. The transformation itself is given in the next section.

Definition 2.2 (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 2.3 (NW-Type PRGs against Nondeterministic Circuits)**.** *An algorithm* $\mathsf{NWPRG} : \{0,1\}^{d(n)} \to \{0,1\}^n$ *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* $\mathsf{NWPRG}(s)$*, 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 [SU01] regarding the existence NW-type PRGs as above assuming worst-case hardness for non-deterministic circuits.

Theorem 2.4 ([SU01])**.** *Assume there exists a function* $f : \{0,1\}^n \to \{0,1\}$ *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 against non-deterministic circuits of size* $t(n)$ $\mathsf{NWPRG} : \{0,1\}^{d(n)} \to \{0,1\}^n$*, where* $d(n) = O(\log n)$*.*

We remark that the above is a worst-case assumption in the sense that the function f needs to be hard in the worst-case (and not necessarily in the average-case). The 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 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 2.5 (BMY-Type PRGs)**.** *An algorithm* $\mathsf{BMYPRG} : \{0,1\}^{d(n)} \to \{0,1\}^n$ *is a BMY-generator if it is computable in time* $\mathrm{poly}(d(n))$ *and any polynomial-size adversary distinguishes* $U \leftarrow \{0,1\}^n$ *from* $\mathsf{BMYPRG}(n)$*, where* $s \leftarrow \{0,1\}^{d(n)}$*, with negligible advantage* $n^{-\omega(1)}$*.*

Theorem 2.6 ([HILL99])**.** *BMY-type pseudo-random generators can be constructed from any one-way function.*

3 The Error-Removing 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 $\alpha = \alpha(\lambda) \leq 2^{-\lambda m - 2}$. 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 the parameters $m, n, \ell, \Pi, f, \mathsf{BMYPRG}$, $t = \lambda^{O(1)}$, $d(\lambda) = O(\log \lambda)$, 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 i-th 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 in turn computes 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 note that had r^{NW} been chosen at truly random (instead of using NWPRG) then for any input, with high probability over the choice of r^{NW}, the corresponding scheme would have been perfectly correct.

Claim 3.1. *For any $x \in \{0,1\}^{n\times m}$,*

$$\Pr_{r^{\mathsf{NW}}\leftarrow\{0,1\}^{\ell\times m}}\left[\exists s_1,\ldots,s_m\in\{0,1\}^{\lambda}: \begin{array}{c} f(x)\neq \Pi(1^{\lambda},x,r) \\ r = r_s^{\mathsf{BMY}}\oplus r^{\mathsf{NW}}\end{array}\right] \leq \frac{1}{4}\ ,$$

where $r_s^{\mathsf{BMY}} = (\mathsf{BMYPRG}(s_1), \ldots, \mathsf{BMYPRG}(s_m))$.

Proof. Fixing any such x and $s = (s_1, \ldots, s_m)$, the string $r = r_s^{\mathsf{BMY}} \oplus r^{\mathsf{NW}}$ is distributed uniformly at random. In this case, the scheme is guaranteed to err with probability at most $\alpha \leq 2^{-\lambda m}/4$. The claim now follows by taking a union bound over all $2^{\lambda m}$ tuples $s_1, \ldots, s_m$. □

We now claim that a similar property holds with roughly the same probability when r^{NW} is pseudorandom as in the actual transformation.

Claim 3.2. *For any $x \in \{0,1\}^{n\times m}$,*

$$\Pr_{j\leftarrow\{0,1\}^{d}}\left[\exists s_1,\ldots,s_m\in\{0,1\}^{\lambda}: \begin{array}{c} f(x)\neq \Pi(1^{\lambda},x,r) \\ r = r_s^{\mathsf{BMY}}\oplus r_j^{\mathsf{NW}}\end{array}\right] \leq \frac{1}{4}+\frac{1}{t}\ ,$$

where $r_s^{\mathsf{BMY}} = (\mathsf{BMYPRG}(s_1), \ldots, \mathsf{BMYPRG}(s_m))$ and $r_j^{\mathsf{NW}} = \mathsf{NWPRG}(j)$.

Proof. Assume towards contradiction that the claim does not hold for some $x \in \{0,1\}^{n\times m}$. We construct a non-deterministic distinguisher that breaks NWPRG. The distinguisher, given r^{NW}, non-deterministically guesses $s_1, \ldots, s_m$, computes $r^{\mathsf{BMY}} = (\mathsf{BMYPRG}(s_1), \ldots, \mathsf{BMYPRG}(s_m))$, $r = r^{\mathsf{NW}} \oplus r^{\mathsf{BMY}}$, and checks whether $f(x) \neq \Pi(1^{\lambda}, x, r)$. As we just proved in the previous claim, when r^{NW} is truly random, such a witness $s_1, \ldots, s_m$ 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 larger than $\frac{1}{t} + \frac{1}{4}$.

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 $t > \max(t', 8)$, meaning that the constructed distinguisher indeed breaks NWPRG. □

With the last claim, we now conclude the proof of Proposition 3.1. Indeed, for any input x, when emulating the k-fold repetition $\Pi_{\otimes k}(1^\lambda, x, r)$, the randomness used for the j-th copy $\Pi(1^\lambda, x, r_j)$ is $r_j = r_j^{\mathsf{NW}} \oplus r_{s_j}^{\mathsf{BMY}}$ where $r_j^{\mathsf{NW}} = \mathsf{NWPRG}(j)$ and $r_{s_j}^{\mathsf{BMY}} = (\mathsf{BMYPRG}(s_{j1}), \ldots, \mathsf{BMYPRG}(s_{j1}))$. By the last claim, for all but a $\frac{1}{4} + \frac{1}{t} \leq \frac{3}{8}$ fraction of the NW-seeds j, any choice of BMY-seeds s_j yields the correct result $y_j = f(x)$ in the corresponding execution $\Pi(1^\lambda, x, r_j)$. In particular, it is always the case that the majority of executions results in $y = f(x)$, as required.

□

Security. We now observe that the randomness generated according to the transformation is indistinguishable from real randomness. Intuitively, this means that if the original scheme was secure under parallel-repetition, when the honest parties use real randomness, it will remain as secure when using randomness generated according to the transformation. Examples are given in the next section.
Concretely, we consider two distributions r^{tra} and r^{uni} on randomness for the parties in $\Pi_{\otimes k}$:

1. In $r^{\mathsf{tra}} = \left(r_{ij}^{\mathsf{tra}} : i \in [m], j \in [k]\right)$, each r_{ij}^{tra} is computed as in the above transformation; namely $r_{ij} = r_{ij}^{\mathsf{BMY}} \oplus r_{ij}^{\mathsf{NW}}$, where $r_{i,j}^{\mathsf{BMY}} = \mathsf{BMYPRG}(s_{ij})$ for a random seed $s_{ij} \leftarrow \{0,1\}^\lambda$ and r_{ij}^{NW} is the i-th ℓ-bit block of $\mathsf{NWPRG}(j)$.

2. In $r^{\mathsf{uni}} = \left(r_{ij}^{\mathsf{uni}} : i \in [m], j \in [k]\right)$, each r_{ij}^{uni} is sampled uniformly at random; namely $r_{ij}^{\mathsf{uni}} \leftarrow \{0,1\}^\ell$.

Proposition 3.2. *r^{tra} and r^{uni} are computationally indistinguishable.*

Proof. By the security of the BMY PRG, for any i, j:

$$r_{ij}^{\mathsf{tra}} = r_{ij}^{\mathsf{BMY}} \oplus r_{ij}^{\mathsf{NW}} = \mathsf{BMYPRG}(s_{ij}) \oplus r_{ij}^{\mathsf{NW}} \approx_c r_{ij}^{\mathsf{uni}} \oplus r_{ij}^{\mathsf{NW}} \equiv r_{ij}^{\mathsf{uni}} \quad .$$

Since r_{ij}^{tra} (respectively r_{ij}^{uni}) is generated independently from all other $r_{i'j'}^{\mathsf{tra}}$ (respectively $r_{i'j'}^{\mathsf{uni}}$), the proposition follows by a standard hybrid argument. □

Removing the Assumption Regarding Tiny Error. Above we assumed that $\alpha(\lambda) \leq 2^{-\lambda m - 2}$. We can start from any $\alpha \leq \frac{1}{2} - \eta$, for $\eta = \lambda^{-O(1)}$, perform $k' = O(\lambda m \eta^{-2})$ 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 for *any* polynomial number of repetitions.

4 Examples of Interest

We now discuss three examples of interest.

Public-Key Encryption. Our first example concerns public-key encryption. We start by recalling the definition.

Definition 4.1 (Public-Key Encryption)**.** *For a message space $\mathcal{M}$, and function $\alpha(\cdot) \leq 1$, a triple of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, where the first two are PPT and third is deterministic polynomial-time, is said to be a* public-key encryption scheme *for $\mathcal{M}$ with $(1-\alpha)$-correctness if it satisfies:*

1. $(1-\alpha)$**-Correctness:** *for any $m \in \mathcal{M}$ and security parameter λ,*

$$\Pr_{\mathsf{Gen},\mathsf{Enc}}\left[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m \;\middle|\; (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)\right] \geq 1 - \alpha(\lambda) \ .$$

2. **Semantic security:** *for any polynomial-size distinguisher $\mathcal{D}$ there exists a negligible function $\mu(\cdot)$, such that for any two messages $m, m' \in \mathcal{M}$ of the same size:*

$$\left|\Pr[\mathcal{D}(\mathsf{Enc}_{pk}(m)) = 1] - \Pr[\mathsf{Enc}_{pk}(m')) = 1]\right| \leq \mu(\lambda) \ ,$$

where the probability is over the coins of Enc *and the choice of pk sampled by* $\mathsf{Gen}(1^\lambda)$*.*

Public-key encryption can be modeled as a three-party scheme Π 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 $\mathsf{Enc}_{pk}(m; r_2)$, 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 $\Pi_{\otimes k}$, the generator $\mathsf{Gen}(1^\lambda; r_{1j})$ is applied k independent times, with fresh randomness r_{1j} for each $j \in [k]$, to generate corresponding keys $pk = \{pk_j\}, sk = \{sk_j\}$. Encryption involves k independent encryptions:

$$\mathsf{Enc}^{\otimes k}_{pk}(m; r_2) := \mathsf{Enc}_{pk_1}(m; r_{21}), \ldots, \mathsf{Enc}_{pk_k}(m; r_{2k}) \ .$$

As defined in Section 3, when applying the error-removal transformation, the randomness $r = (r_{ij} : i \in [2], j \in [k])$ is sampled according to r^{tra} instead of truly at random according to r^{uni}. Decryption is done by decrypting each encryption with the corresponding sk_j and outputting the majority.

The correctness of the new scheme given by the transformation, follows as in Proposition 3.1. We next observe that the new scheme is also secure. Concretely, for any (infinite sequence of) two messages $m, m' \in \mathcal{M}$,

$$\mathsf{Enc}^{\otimes k}_{pk}(m; r_2^{\mathsf{tra}}) \approx_c \mathsf{Enc}^{\otimes k}_{pk}(m; r_2^{\mathsf{uni}}) \approx_c \mathsf{Enc}^{\otimes k}_{pk}(m'; r_2^{\mathsf{uni}}) \approx_c \mathsf{Enc}^{\otimes k}_{pk}(m'; r_2^{\mathsf{tra}}) \ .$$

The fact that $\mathsf{Enc}^{\otimes k}_{pk}(m; r_2^{\mathsf{uni}}) \approx_c \mathsf{Enc}^{\otimes k}_{pk}(m'; r_2^{\mathsf{uni}})$ follows from the semantic security of the underlying encryption scheme and a standard hybrid argument. The first and last indistinguishability relations follow from the fact that $r_2^{\mathsf{tra}} \approx_c r_2^{\mathsf{uni}}$ (by Proposition 3.2).

In [DNR04b], 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. For the specific case

Indistinguishability Obfuscation. Our second example concerns indistinguishability obfuscation (IO) [BGI+12]. We start by recalling the definition.

Definition 4.2 (Indistinguishability Obfuscation)**.** *For a class of circuits $\mathcal{C}$, and function $\alpha(\cdot) \leq 1$, a PPT algorithm $\mathcal{O}$ is said to be an* indistinguishability obfuscator *for $\mathcal{C}$ with $(1-\alpha)$-correctness if it satisfies:*

1. $(1-\alpha)$**-Correctness:** *for any $C \in \mathcal{C}$ and security parameter λ,*

$$\Pr_{\mathcal{O}}\left[\forall x : \mathcal{O}(C, 1^\lambda)(x) = C(x)\right] \geq 1 - \alpha(\lambda) \ .$$

2. **Indistinguishability:** *for any polynomial-size distinguisher $\mathcal{D}$ there exists a negligible function $\mu(\cdot)$, such that for any two circuits $C, C' \in \mathcal{C}$ that compute the same function and are of the same size:*

$$\left|\Pr[\mathcal{D}(\mathcal{O}(C, 1^\lambda)) = 1] - \Pr[\mathcal{D}(\mathcal{O}(C', 1^\lambda)) = 1]\right| \leq \mu(\lambda) \ ,$$

where the probability is over the coins of $\mathcal{D}$ and $\mathcal{O}$.

IO can be modeled as a two-party scheme Π 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 obfuscated circuit $\widetilde{C} = \mathcal{O}(C, 1^\lambda; r_1)$, which is sent to the evaluator. The evaluator has an input x for the circuit, and no randomness, it computes $\widetilde{C}(x)$ and outputs the result. (In this case the function computed by Π is $f(C, x) = (\perp, C(x))$.)

In the repeated scheme $\Pi_{\otimes k}$, obfuscation involves k independent obfuscations:

$$\mathcal{O}^{\otimes k}(C, 1^\lambda; r_1) := \mathcal{O}(C, 1^\lambda; r_{11}), \ldots, \mathcal{O}(C, 1^\lambda; r_{1k}) \ .$$

As defined in Section 3, when applying the error-removal transformation, the randomness $r = (r_{1j} : j \in [k])$ is sampled according to r^{tra} instead of truly at random according to r^{uni}. Evaluation for input x is done by running each obfuscated circuit on the input x and outputting the majority of outputs.

The correctness of the new scheme given by the transformation, follows as in Proposition 3.1. We now observe that the new scheme is also secure, which follows similarly to the case of public-key encryption considered above. Concretely, for any (infinite sequence of) two equal-size circuits $C, C' \in \mathcal{C}$,

$$\mathcal{O}^{\otimes k}(C, 1^\lambda; r_1^{\mathsf{tra}}) \approx_c \mathcal{O}^{\otimes k}(C, 1^\lambda; r_1^{\mathsf{uni}}) \approx_c \mathcal{O}^{\otimes k}(C', 1^\lambda; r_1^{\mathsf{uni}}) \approx_c \mathcal{O}^{\otimes k}(C', 1^\lambda; r_1^{\mathsf{tra}}) \ .$$

The fact that $\mathcal{O}^{\otimes k}(C, 1^\lambda; r_1^{\mathsf{uni}}) \approx_c \mathcal{O}^{\otimes k}(C', 1^\lambda; r_1^{\mathsf{uni}})$ follows from the security of the underlying obfuscation scheme and a standard hybrid argument. The first and last indistinguishability relations follow from the fact that $r_1^{\mathsf{tra}} \approx_c r_1^{\mathsf{uni}}$ (by Proposition 3.2).

In [BV16], Bitansky and Vaikuntanathan show how indistinguishability obfuscation [BGI+12] where the obfuscated circuit may err also on a large fraction of *inputs* can be transformed into one that only has a tiny error over the randomness of the obfuscator as required here. Applying our transformation, we can further turn such schemes into perfectly correct ones.

MPC. Our third and last example concerns multi-party computation (MPC) protocols. There are several models for capturing the adversarial capabilities in an MPC protocol. Roughly speaking, our transformation can be applied whenever the protocol is secure against parallel repetition. In the new protocol, perfect correctness will be guaranteed when all the parties behave honestly. The

security guarantee given by the new protocol will be inherited from the original repeated protocol. We stress that, in the case of corrupted parties, the transformed protocol does not provide any correctness guarantees beyond those given by the original (repeated) protocol. In particular, if the adversary can inflict a certain correctness error in the original (repeated) protocol, it may also be able to do so in the transformed protocol.

We now give more details. Since we rely on standard MPC conventions, we shall keep our description relatively light (for further reading, see for instance [Can01, Gol04]). We consider protocols with security against *static corruptions* according to the *real-ideal paradigm*. For simplicity of exposition, we restrict attention to the single-execution setting. (Later, we explain how the transformation can also be applied in the setting of multiple executions, for example, in the UC model [Can01].) In this setting, the adversary $\mathcal{A}$ corrupts some set of parties $C \subseteq [m]$, which it fully controls throughout the protocol, and can also choose the inputs for honest parties at the onset of the computation. The *adversarial view* in the protocol consists of all the communication generated by the honest parties and their respective outputs. We denote by $\mathsf{Real}^{\mathcal{A}}_{\Pi}(1^\lambda, z; r)$ the polynomial-time process that generates the adversarial view and the outputs of the honest parties in $[m] \setminus C$ when these parties execute protocol Π for functionality f with randomness $r = (r_{i_1}, \ldots, r_{i_{m-|C|}})$, and a PPT adversary $\mathcal{A}$ with auxiliary input z controlling the parties in C.

The requirement is that the output of this process can be simulated by a PPT process $\mathsf{Ideal}^{\mathcal{S}}_{f}(1^\lambda, z)$ called the ideal process where $\mathcal{A}$ is replaced by an efficient *simulator* $\mathcal{S}$. The simulator can only submit inputs $x_1, \ldots, x_m$ to f, learn the outputs of the corrupted parties in C, and has to generate the adversarial view. The ideal process outputs the view generated by the simulator as well as the output generated by f for the honest parties.

As before, we denote by $\Pi_{\otimes k}$ the k-fold parallel repetition of a protocol Π for computing $f_{\otimes k}(x) = (f(x))^k$, where each honest party $i \in [m] \setminus C$, given input x_i, runs k parallel copies of Π, all with the same input x_i and obtains outputs $y_{i1}, \ldots, y_{ik}$. We consider protocols that are secure under parallel repetition in the following sense.

Definition 4.3. *We say that an MPC protocol Π (for some functionality f) is secure under parallel repetition with respect to an ideal process Ideal if for any PPT adversary $\mathcal{A}$ and polynomial $k(\lambda)$ there exists a PPT simulator $\mathcal{S}$ such that for any (infinite sequence of) security parameter $\lambda \in \mathbb{N}$ and auxiliary input in $z \in \{0,1\}^*$,*

$$\mathsf{Real}^{\mathcal{A}}_{\Pi_{\otimes k}}(1^\lambda, z) \approx_c \mathsf{Ideal}^{\mathcal{S}}_{f_{\otimes k}}(1^\lambda, z) \ .$$

We denote by Π^{tra} the protocol Π for computing f after applying the transformation from Section 3 where Π is repeated in k times in parallel, the randomness of parties is derived as defined in the transformation, and the final output of party i is set to $\mathsf{majority}(y_{i1}, \ldots, y_{ik})$. When all the parties act honestly, the correctness of the new protocol Π^{tra} given by the transformation, follows as in Proposition 3.1.

We show that if the original protocol is secure under parallel repetition then the transformed protocol is as secure.

Claim 4.4. *Assume that Π is a protocol for f that is secure under parallel repetition (in the sense of Definition 4.3). For any PPT adversary $\mathcal{A}$ against Π^{tra}, viewing $\mathcal{A}$ as an adversary against $\Pi_{\otimes k}$, let $\mathcal{S}$ be its simulator given by Definition 4.3. Then for any (infinite sequence of) security parameter λ, and auxiliary input z,*

$$\mathsf{Real}^{\mathcal{A}}_{\Pi^{\mathsf{tra}}}(1^\lambda, z) \approx_c \mathsf{Ideal}^{\mathcal{S}}_{f}(1^\lambda, z) \ .$$

Proof. Let $\Pi^{\mathsf{maj}}_{\otimes k}$ be the protocol where the parties first execute the k-fold repetition of $\Pi_{\otimes k}$ and then each party sets its final output to be the majority of the outputs obtained in that execution. Then we first note that

$$\mathsf{Real}^{\mathcal{A}}_{\Pi^{\mathsf{tra}}}(1^\lambda, z) \equiv \mathsf{Real}^{\mathcal{A}}_{\Pi^{\mathsf{maj}}_{\otimes k}}(1^\lambda, z; r^{\mathsf{tra}}) \ ,$$

where r^{tra} is the randomness of the honest parties, generated according to our transformation. By Proposition 3.2, it holds that:

$$\mathsf{Real}^{\mathcal{A}}_{\Pi^{\mathsf{tra}}}(1^\lambda, z) \equiv \mathsf{Real}^{\mathcal{A}}_{\Pi^{\mathsf{maj}}_{\otimes k}}(1^\lambda, z; r^{\mathsf{tra}}) \approx_c \mathsf{Real}^{\mathcal{A}}_{\Pi^{\mathsf{maj}}_{\otimes k}}(1^\lambda, z; r^{\mathsf{uni}}) \ ,$$

where r^{tra} is randomness generated according to our transformation and r^{uni} is truly random. It is left to note that

$$\mathsf{Real}^{\mathcal{A}}_{\Pi^{\mathsf{maj}}_{\otimes k}}(1^\lambda, z; r^{\mathsf{uni}}) \approx_c \mathsf{Ideal}^{\mathcal{S}}_{f}(1^\lambda, z) \ .$$

Indeed, recall that by Definition 4.3,

$$\mathsf{Real}_{\Pi_{\otimes k}}{}^{\mathcal{A}}(1^\lambda, z; r^{\mathsf{uni}}) \approx_c \mathsf{Ideal}^{\mathcal{S}}_{f_{\otimes k}}(1^\lambda, z) \ ,$$

and each of the first two distributions can be efficiently computed from the respective distribution in the second two, by fixing the (single) output of each honest party to be the majority of its outputs. □

Applying the Transformation in More General Models. Above, we have considered a model with a single execution. The analysis naturally extends to more general models such as the model of universally composable (UC) protocols [Can01], where multiple executions controlled by an adversarial *environment* can be performed. Indeed, the only feature of the model we have relied on is that the real world view can be generated using the randomness of honest parties as external input (regardless of how the randomness was generated), which is the case as long corruptions are static, and the adversary is never exposed to the randomness of honest parties, but only to the communication between parties. This is also the case in the UC model.

Acknowledgements

We thank Stefano Tessaro for pointing out [HR05] and [LT13]. We also thank the reviewers of EUROCRYPT 2017 for their valuable comments.

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.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 505–514. ACM, 2014.

[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.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 401–427. Springer, 2015.

[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.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[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. *Advances in Computing Research: A Research Annual (Randomness and Computation, S. Micali, ed.)*, 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.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[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.

[HR05] Thomas Holenstein and Renato Renner. One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 478–493. Springer, 2005.

[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.

[LT13] Huijia Lin and Stefano Tessaro. Amplification of chosen-ciphertext security. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 503–519. Springer, 2013.

[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.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il