J. Cryptol. (2009) 22: 75–92
DOI: 10.1007/s00145-008-9029-7

Journal of
CRYPTOLOGY

Chernoff-Type Direct Product Theorems

Russell Impagliazzo[1] , Ragesh Jaiswal[2] , and Valentine Kabanets
University of California San Diego, San Diego, USA
russell@cs.ucsd.edu
and
Institute of Advanced Studies, Princeton, USA
and
University of California San Diego, San Diego, USA
rjaiswal@cs.ucsd.edu
and
Simon Fraser University, Vancouver, Canada
kabanets@cs.sfu.ca

Received 8 November 2007 and revised 26 August 2008
Online publication 24 September 2008

Abstract. Consider a challenge-response protocol where the probability of a correct response is at least α for a legitimate user and at most $\beta < \alpha$ for an attacker. One example is a CAPTCHA challenge, where a human should have a significantly higher chance of answering a single challenge (e.g., uncovering a distorted letter) than an attacker; another example is an argument system without perfect completeness. A natural approach to boost the gap between legitimate users and attackers is to issue many challenges and accept if the response is correct for more than a threshold fraction, for the threshold chosen between α and β. We give the first proof that parallel repetition with thresholds improves the security of such protocols. We do this with a very general result about an attacker's ability to solve a large fraction of many independent instances of a hard problem, showing a Chernoff-like convergence of the fraction solved incorrectly to the probability of failure for a single instance.

Key words. Challenge-response protocols, Parallel repetition with threshold, Direct product theorem.

1. Introduction

Cryptographic protocols require problems that are easy for legitimate users but hard for attackers. The hardness of a problem may be either computational (when attackers are assumed computationally bounded) or information-theoretic (when attackers are computationally unbounded). Ideally, a problem should be reliably easy for legitimate users

[1] Research partially supported by NSF Awards CCR-0515332 and CNS-0716790. Views expressed are not endorsed by the NSF.

[2] Research partially supported by NSF Awards CCR-0515332, CCF-0634909, CNS-0524765 and CNS-0716790. Views expressed are not endorsed by the NSF.

(i.e., the chance of failure for legitimate users should be negligible) but reliably hard for attackers (i.e., the chance of the attacker's success is negligible). In reality, one may have a problem which is only somewhat easier for legitimate users than for attackers, i.e., the gap between the ability of legitimate users to solve the problem and that of attackers is relatively small. It is thus important to have a method for increasing this gap, thereby improving the security of cryptographic protocols based on such problems.

Direct product theorems provide one such method for making problems reliably hard for attackers. The idea is that if an attacker has some chance of failing on a single challenge, the chance of solving multiple independent challenges should drop exponentially fast with the number of challenges. Examples of such theorems in cryptography include Yao's theorem that weak one-way functions imply strong one-way functions [12] and the results of [1,2] showing similar drops even when an attacker cannot know for certain whether a response to a challenge is correct. Direct product theorems are also important in average-case complexity, circuit complexity, and derandomization. While intuitive, such results are frequently nontrivial to establish. Moreover, there are settings where the intuition is incorrect, and many instances are not proportionally harder; examples where direct products fail are parallel repetition for multiple round protocols and for nonverifiable puzzles [1,2,10].

A standard direct product theorem can only be used to amplify the gap between legitimate users and attackers if legitimate users are successful with high probability. Indeed, the legitimate user's chance of solving k independent challenges also drops exponentially fast with k. So unless the legitimate user's probability of failure is not much more than $1/k$ to start, both legitimate users and attackers will almost certainly fail to solve all of the problems.

Suppose that a legitimate user has probability α of solving a randomly generated challenge, while an attacker has probability $\beta < \alpha$. For k independent random challenges, we expect the legitimate user to be correct on αk of them. By Chernoff bounds, the actual number of correct answers will be very close to αk with high probability. On the other hand, the expected number of correct answers by the attacker is βk. Intuitively, it should be unlikely that the actual number of correct answers is much larger than the expected number. That is, by analogy with Chernoff bounds, the attacker's probability of answering correctly on significantly more that the expected number βk of random k challenges should be exponentially small in the expectation βk.

This intuition turns out to be correct. The main result of our paper is such a strengthening of the direct product theorem for a very general class of problems, *weakly verifiable puzzles*, introduced in [2].

1.1. *Example: CAPTCHA*

Before defining the class of weakly verifiable puzzles, we consider an example of a cryptographic protocol where our results apply. A CAPTCHA protocol is meant to distinguish between humans and programs, usually using a visual challenge based on distorted text with extraneous lines [11]. While there seems to be a large gap between the abilities of typical humans and the best current vision algorithms to solve these challenges, algorithms can solve a non-negligible fraction of the puzzles, and many humans (including us) fail a non-negligible fraction of the puzzles.

An obvious, intuitive way to increase the gap is to issue many independent challenges and accept if the solver is successful on a larger fraction than expected for an attacker, even if the solver does not succeed on all challenges. The fact that *sequential* repetition improves the gap was observed by [11]. The authors of [11] also imply that *parallel* repetition improves the gap, referring to the results in [1] for this "more complicated" case. Indeed, the direct product theorem of [1] (improved by [2]) does apply to parallel repetition of CAPTCHA protocols, but it only shows that the probability of algorithmic success decreases with repetitions, not that the gap improves. Our stronger version of the direct product theorem gives the first proof that the parallel repetition protocol suggested in [11] does indeed improve the gap between legitimate users and attackers.

A CAPTCHA protocol issues a puzzle (e.g., distorted text) such that the correctness of a solution to the puzzle is easy to verify by the generator of the puzzle (who knows the text that was distorted), but not by the attacker (who is just given the puzzle, not the way it was generated). Such puzzles are called *weakly verifiable* in [2].

1.2. *Our Main Result*

Before stating our main theorem, we need several definitions. For a weakly verifiable puzzle P and a natural number k, we denote by P^k the k-wise direct product of P, i.e., P^k is the puzzle that asks k independent challenges from P. For a parameter $0 \leq \delta \leq 1$, we say that P is δ-*hard* for time t if every randomized algorithm running in time $t(n)$ has probability at least δ of answering incorrectly a randomly generated challenge from P (where the probability is both over input challenges from P and the internal randomness of the algorithm), for sufficiently large input size n. Finally, for parameters $k \in \mathbb{N}$ and $0 \leq \nu, \delta \leq 1$, we say that the k-wise direct product puzzle P^k is ν-*approximately* δ-*hard* for time t if every randomized algorithm running in time $t(n)$ has probability at least δ (over k-tuples of challenges from P^k, and its internal randomness) of answering incorrectly at least νk of the input k challenges.

Our main theorem states that for δ-hard puzzle P, its k-wise direct product P^k is, essentially, δ-approximately $(1 - o(1))$-hard. That is, not only is it impossible to solve all k challenges for a non-negligible fraction of k-tuples from P^k, but also it is impossible to make significantly fewer than the expected number δk of mistakes on the input k challenges. More precisely, we have the following.[1]

Theorem 1.1 (Main Theorem). *Let P be a weakly verifiable puzzle that is δ-hard for time t. Let $k \in \mathbb{N}$ and $\gamma > 0$ be arbitrary, and let $\epsilon \geq (100/\gamma\delta) \cdot e^{-\gamma^2 \delta k/40}$. Then the direct product puzzle P^k is $(1 - \gamma)\delta$-approximately $(1 - \epsilon)$-hard for time $t' = t(n) \cdot \mathrm{poly}(\epsilon, 1/n, 1/k)$.*

We call this a *Chernoff-type* direct product theorem, since it shows that the "tail bound" on the number of correctly solved puzzles drops exponentially in the region beyond its expectation.

[1] The parameters in the Main Theorem stated here are stronger than those reported in the conference version of this paper [6]; the improvement comes from using the sampling lemmas of [7] instead of those from [5].

Standard Chernoff bounds show that, if the legitimate user can solve the problem with probability of failure less than, say, $(1 - 2\gamma)\delta$, then they will succeed in solving all but $(1 - \gamma)\delta k$ of the input k challenges, for almost all k-tuples from P^k. Thus our Chernoff-type direct product theorem indeed provides a way to amplify any gap between legitimate users and attackers.

Finally, we should also note that for direct products with threshold, it is impossible to get the bound $\epsilon = (1 - \delta)^k$, which is possible for standard direct products [2]. Indeed, consider the case of a puzzle P such that P is easy for $(1 - \delta)$ fraction of inputs but is information-theoretically impossible to solve on the remaining δ fraction of inputs. Then the probability of making fewer than δk mistakes on a given random k-tuple of challenges is the tail bound for the binomial distribution where one flips k independent coins with the "heads" probability δ. When δk is sufficiently far from 0 and far from k (e.g., for constant $0 < \delta < 1$), then the Chernoff bound provides a tight estimate for this tail bound. Thus the bound of our main theorem cannot be significantly improved, except possibly for making the constant in the exponent of ϵ in Theorem 1.1 (currently 40) closer to that of the Chernoff bound (which can be as low as 2).

1.3. *Weakly Verifiable Puzzles: Definition and Examples*

Our result holds for weakly verifiable puzzles defined by [2]. A *weakly verifiable puzzle* has two components:

- a distribution ensemble $D = \{D_n\}_{n \geq 1}$ on pairs (x, α), where x is called the puzzle and α the check string (n is the security parameter); and
- a polynomial-time computable relation $R((x, \alpha), y)$, where y is a string of a fixed polynomially-related length.

The puzzle is thought of as defining a type of challenge x, with y being the solver's response. However, the correctness of the response is not easily verified (and may not be well defined) given just x. On the other hand, the party generating the puzzle x also knows α, so can verify correctness.

In [2], the distribution D is restricted to be polynomial-time sampleable. In this case, without loss of generality, we can assume that α is the n-bit random tape used to generate the puzzle and check string (if not, we can redefine R as R' which first generate the check string from the random tape, then verifies R). Thus, to simplify the notation in our proofs, we usually assume that α is a uniformly generated n-bit string and that x is a function of α. A version of our result also holds when D is not polynomial-time sampleable, but only for nonuniform adversaries (since many samples from D are required as advice).

Here we summarize some important properties of weakly verifiable puzzles. The generation and verification procedures for the puzzles are polynomial-time algorithms. A puzzle may have multiple correct answers (since an answer to a puzzle is verified using a relation). The same puzzle may be generated using multiple random tapes; we call such puzzles *ambiguous*. Moreover, since the verification procedure takes as input the random tape α used to generate a puzzle x, the set of correct answers for x depends on α, and these sets of corrects answers may be different (even disjoint) for different random tapes α and α' that generate the same puzzle x.

Some examples of how weakly verifiable puzzles arise in different settings include:

1. A challenge-response protocol where a prover is trying to get a verifier to accept them as legitimate (e.g., a CAPTCHA protocol where the prover is trying to convince the verifier to accept them as human). We assume that the verifier is polynomial-time with no secret inputs (although an honest prover may have secret inputs). Let α be the random bits used by the verifier. In the first round, the verifier sends a challenge $x = g(\alpha)$, and the prover sends a response y. The verifier then decides whether to accept by some polynomial time algorithm $R(\alpha, y)$. Our results are interesting if there is some chance that the honest prover will be rejected, such as an honest human user failing a CAPTCHA challenge based on visual distortion.
2. A secret-agreement protocol with a passive eavesdropper. Let r_A be the random tape used by one party, and r_B that by the other party. Then the conversation C is a function of both r_A, r_B, as is the message m agreed upon. The eavesdropper succeeds if she computes m given C. Then consider $\alpha = (r_A, r_B)$, $x = C$, and $R(C, (r_A, r_B), y)$ if y is the message agreed upon by the two parties using r_A and r_B. Note that there may be some tapes where the parties fail to agree and thus have no success. Our result shows that, if the parties agree more probably than the eavesdropper can guess the secret, then by running the protocol several times they will almost certainly have more shared secrets than the eavesdropper can guess. Note that, unlike for challenge-response protocols, here there is no restriction on the amount of interaction between the legitimate parties (as long as the eavesdropper is passive).
3. Let f be a (weak) one-way function, and b a (partially-hidden) bit for f, in the sense that it is sometimes hard to predict b from $x = f(z)$. Since f may not be one-to-one, b may be hard to predict for either information-theoretic or computational reasons. Here, we let $\alpha = z$, $x = f(\alpha)$, and $R(x, \alpha, b')$ if $b' = b(\alpha)$. Our results say that no adversary given an n-tuple of $x_i = f(z_i)$'s can produce a string closer in relative Hamming distance to $b(\alpha_1) \ldots b(\alpha_n)$ than the hardness of prediction.
4. In the nonuniform setting, our results apply to any function. If f is a function (possibly non-Boolean, or even multi-valued, as long as it has at most a polynomial number of values), we can define α to be (the set of all elements in) $f(x)$. Then $y \in f(x)$ if and only if $y \in \alpha$, so this is testable in polynomial-time given α. This distribution is not necessarily polynomial-time sampleable, so our results would only apply for nonuniform adversaries (e.g., Boolean circuits).

Note that in some examples, success may be ill defined, in that x may not uniquely determine α, and so it may not be information-theoretically possible to know whether $R((x, \alpha), y)$ given only x.

1.4. *Related Work*

The notion of a Direct Product Theorem, where solving multiple instances of a problem simultaneously is proven harder than a single instance, was introduced by Yao in [12]. Due to its wide applicability in cryptography and computational complexity, a number of different versions and proofs of such theorems can be found in the literature; see, e.g., [3] for a good compilation of such results.

In this paper, we use some of the proof techniques (namely the *trust halving strategy*) introduced by Impagliazzo and Wigderson in [8]. Such techniques were also used to prove a version of the Direct Product Theorem in a more general cryptographic setting by Bellare, Impagliazzo, and Naor in [1]. It is shown in [1] that the soundness error decreases exponentially with parallel repetition in any 3-round challenge-response protocol, but such error amplification might not be possible for a general (> 3)-round protocol. Pietrzak and Wikstrom in [10] extend this negative result. On the positive side, Canetti, Halevi, and Steiner in [2] used ideas from [1] to define a general class of *weakly verifiable puzzles* for which they show that parallel repetition amplifies hardness, also giving a quantitative improvement over [1]. More recently, Pass and Venkitasubramaniam [9] show similar positive results for constant-round public-coin protocols.

All the previous results mentioned above consider parallel repetition without threshold, i.e., they consider the hardness of answering *all* the questions.

Comparing the Techniques of [1] and Those of [2] Our construction uses a version of the trust-reducing strategy from [1,8]. In a trust-reducing strategy, the input puzzle is hidden among $(k-1)$ randomly generated puzzles, and the number of mistakes the attacker makes on the random puzzles is used to compute the probability with which the algorithm trusts the attacker's answer for the input puzzle.

A different approach was used in [2]. Their proof strategy (similar to that of Goldreich, Nisan, and Wigderson [3]) is roughly as follows. Suppose that some attacker $\bar{C}$ correctly answers all k challenges for at least ϵ fraction of k-tuples from some direct product puzzle P^k, where P is δ-hard. Then (arguing by induction) one shows that there exists a position $1 \leq i \leq k$ and fixed inputs $x_1, \ldots, x_{i-1}$ for the positions before i such that the probability of getting a correct answer for the ith input in a given k-tuple, conditioned on the attacker's answers for the positions $i+1, \ldots, k$ being correct, is at least $1-\delta$. Thus, to answer a challenge x, one places x into position i, randomly generates challenges for the positions higher than i, runs the attacker $\bar{C}$ on the constructed k-tuple, and outputs the answer of $\bar{C}$ for x if the answers of $\bar{C}$ on *all* the $(k-i)$ random challenges are correct; otherwise one repeats with new $(k-i)$ random challenges.

The argument of [2] allows one to conclude that $\epsilon = (1-\delta)^k$, which is information-theoretically the best possible bound, is a quantitative improvement on the bound on ϵ shown in [1]. While the techniques of [2] yield stronger (optimal) bounds for the direct product than those of [1], we do not see how to use the techniques of [2] for the case of direct products *with threshold* that we consider in the present paper. In our case, we need to deal with a variable number of mistakes even for "good" k-tuples, and we manage to adapt the techniques of [1] to handle such mistakes.

1.5. *Our Techniques*

As in [1,8], our proof of the main theorem is constructive: we show how to use a breaking strategy that solves the threshold puzzle with probability ϵ as a subroutine in an algorithm that solves a single puzzle with probability greater than $1-\delta$. However, we need to deviate substantially from the previous analysis.

The way it is argued in [1,8] that all but δ fraction of inputs are easy is as follows. Suppose an algorithm A succeeds on a significant fraction of k-tuples of random puzzle instances. Then one constructs another algorithm A' such that, for every subset of puzzle

instances H of density at least δ, algorithm A' succeeds almost surely on a random instance in H. Now consider the set of all puzzle instances where A' gives a wrong answer. By the above, this set must have density less than δ (or else A' would succeed almost surely on a random element in the set).

In contrast, in the threshold scheme, it is not possible to construct an algorithm A' with the similar guarantee that A' succeeds almost surely on a random instance in H, for every subset H of density at least δ. Indeed, for a given puzzle P, there may be a subset H' of density $(1-\gamma)\delta$ of input instances where no algorithm can succeed with non-negligible probability, while all the other instances outside H' are easy to solve. In this case, there is an algorithm that solves almost all k-tuples of puzzle instances if we allow up to about $(1-\gamma)\delta k$ errors. However, for any set H of density δ such that $H' \subseteq H$, no algorithm A' can succeed on more than γ fraction of elements of H.

In order to get around this obstacle, we need a more *global* way of analyzing the trust-reducing strategy. Our main tools for doing this are sampling lemmas from [7]. The high-level idea is as follows. Let G be the set of k-tuples of puzzle instances where some algorithm A is correct in all but $(1-\gamma)\delta k$ positions. Suppose that G has density ϵ. The trust-reducing strategy essentially allows us to construct an efficient oracle for testing membership in G. The overall strategy for solving a puzzle instance x is then to sample random k-tuples containing x, until getting the tuple that falls into G; for such a tuple, we output the value A gives for the xth position in the tuple.

Since G has density ϵ, we are almost sure to sample a tuple from G within $\text{poly}(1/\epsilon)$ iterations. We use a sampling lemma to argue that, conditioned on sampling a random k-tuple from G, the position of the input x is distributed almost uniformly within the tuple. Hence, in that case, we get the correct answer for x with probability at least $1-(1-\gamma)\delta = 1-\delta+\gamma\delta$ (since every tuple in G has at most $(1-\gamma)\delta k$ bad positions). Accounting for possible errors of our membership oracle for G, the probability of our sampling procedure missing the set G, and the fact that the xth positions is only almost uniform within the tuple, we conclude that our algorithm succeeds on at least $1-\delta$ fraction of inputs x.

2. Preliminaries

2.1. *Basics*

For a natural number k, we will denote by $[k]$ the set $\{1, \ldots, k\}$.

Lemma 2.1 (Hoeffding bound). *Let $X_1, \ldots, X_t$ be independent identically distributed random variables taking values in the interval $[0,1]$, with expectation μ. Let $\chi = (1/t)\sum_{i=1}^{t} X_i$. For any $0 < \nu \leq 1$, we have* $\mathbf{Pr}[\chi < (1-\nu)\mu] < e^{-\nu^2\mu t/2}$.

2.2. *Samplers*

We will consider bipartite graphs $G = G(L \cup R, E)$ defined on a bipartition $L \cup R$ of vertices; we think of L as left vertices, and R as right vertices of the graph G. We allow graphs with multiple edges. For a vertex v of G, we denote by $N_G(v)$ the multiset of its neighbors in G; if the graph G is clear from the context, we will drop the subscript and

simply write $N(v)$. Also, for a vertex x of G, we denote by E_x the set of all edges in G that are incident to x. We say that G is *bi-regular* if the degrees of vertices in L are the same and the degrees of vertices in R are the same.

Let $G = G(L \cup R, E)$ be any bi-regular bipartite graph. For a function $\lambda : [0, 1] \times [0, 1] \to [0, 1]$, we say that G is a λ-*sampler* if, for every function $F : L \to [0, 1]$ with the average value $\mathbf{Exp}_{x \in L}[F(x)] \geq \mu$ and any $0 < \nu < 1$, there are at most $\lambda(\mu, \nu) \cdot |R|$ vertices $r \in R$ where $\mathbf{Exp}_{y \in N(r)}[F(y)] \leq (1 - \nu)\mu$.

We will use the following properties of samplers (proved in [7] for the special case of $\nu = 1/2$); for completeness, we state them with the proofs. The first property says that for any two large vertex subsets W and F of a sampler, the fraction of edges between W and F is close to the product of the densities of W and F.

Lemma 2.2 ([7]). *Suppose $G = G(L \cup R, E)$ is a λ-sampler. Let $W \subseteq R$ be any set of measure at least τ, and let $V \subseteq L$ be any set of measure at least β. Then, for all $0 < \nu < 1$ and $\lambda_0 = \lambda(\beta, \nu)$, we have $\mathbf{Pr}_{x \in L, y \in N(x)}[x \in V \ \& \ y \in W] \geq \beta(1 - \nu)(\tau - \lambda_0)$, where the probability is for the random experiment of first picking a random node $x \in L$ uniformly at random and then picking a uniformly random neighbor y of x in the graph G.*

Proof. We need to estimate the probability of picking an edge between V and W in a random experiment where we first choose a random $x \in L$ and then its random neighbor y. Since the graph G is assumed to be bi-regular, this probability remains the same in the experiment where we first pick a random $y \in R$ and its random neighbor $x \in N(y)$. The latter is easy to estimate using the sampling property of the graph G as follows. Consider the function $F : L \to [0, 1]$ defined as

$$F(x) = \begin{cases} 1 & \text{if } x \in V; \\ 0 & \text{otherwise.} \end{cases}$$

Since the measure of V is at least β, we have $\mathbf{Exp}_{x \in L}[F(x)] \geq \beta$. Let $W' \subseteq W$ be the subset of vertices that have at least $(1 - \nu)\beta$ fraction of their neighbors in V. In other words, W' contains those vertices $r \in R$ such that $\mathbf{Exp}_{y \in N(r)}[F(y)] \geq (1 - \nu)\beta$. Since G is a λ-sampler and W is of measure at least τ, we get that W' is of measure at least $\tau - \lambda_0$. Then, conditioned on picking a vertex $y \in W'$, the probability that its random neighbor is in V is at least $(1 - \nu)\beta$. The lemma follows. □

The second property deals with edge-colored samplers. It basically says that removing some subset of right vertices of a sampler yields a graph which (although not necessarily bi-regular) still has the following property: Picking a random left node and then picking its random neighbor induces roughly the same distribution on the edges as picking a random right node and then its random neighbor.

Lemma 2.3 ([7]). *Suppose $G = G(L \cup R, E)$ is a λ-sampler with the right degree D. Let $W \subseteq R$ be any subset of density at least τ, and let $G' = G(L \cup W, E')$ be the induced subgraph of G (obtained after removing all vertices in $R \setminus W$) with the edge set E'. Let $Col : E' \to \{red, green\}$ be any coloring of the edges of G' such that at most $\eta D|W|$*

edges are colored red for some $0 \le \eta \le 1$. *Then, for all* $0 < \nu, \beta < 1$ *and* $\lambda_0 = \lambda(\beta, \nu)$, *we have*

$$\mathbf{Pr}_{x \in L, y \in N_{G'}(x)}\big[Col(\{x, y\}) = red\big] \le \max\big\{\eta/\big((1-\nu)(1-\lambda_0/\tau)\big), \beta\big\},$$

where the probability is for the random experiment of first picking a uniformly random node $x \in L$ *and then picking a uniformly random neighbor* y *of* x *in the graph* G'.

Proof. For every $x \in L$, let d_x be the degree of x in G', and let $\xi(x)$ be the fraction of red edges incident to x in G'. The probability we want to estimate is exactly $\mu = \mathbf{Exp}_{x \in L}[\xi(x)]$. If $\mu \le \beta$, then we are done. So for the rest of the proof, we will suppose that $\mu > \beta$.

Let $W' \subseteq W$ be the subset of those vertices w where $\mathbf{Exp}_{x \in N(w)}[\xi(x)] \ge (1-\nu)\mu$. (Here we use $N(w)$ to denote the neighborhood $N_{G'}(w)$ of w in G', which is the same as $N_G(w)$ by the definition of G'.) Since G is a λ-sampler and W has measure at least τ in R, we get that W' has measure at least $1 - \lambda_0/\tau$ in W. Hence, we have

$$\sum_{y \in W} \mathbf{Exp}_{x \in N(y)}\big[\xi(x)\big] \ge \sum_{y \in W'} \mathbf{Exp}_{x \in N(y)}\big[\xi(x)\big] \ge |W|(1-\lambda_0/\tau)(1-\nu)\mu. \quad (1)$$

On the other hand, $\sum_{y \in W}(D \cdot \mathbf{Exp}_{x \in N(y)}[\xi(x)])$ is simply the summation over all edges (x, y) in G' where each edge (x, y) with $x \in L$ contributes $\xi(x)$ to the sum. Since the degree of each x is d_x, each $x \in L$ contributes exactly $d_x \xi(x)$, which is the number of incident red edges at x. Hence, the total sum is exactly the number of red edges in G', which is at most $\eta D|W|$ by assumption. It follows that

$$\sum_{y \in W} \mathbf{Exp}_{x \in N(y)}\big[\xi(x)\big] = (1/D)\sum_{x \in L} d_x \xi(x) \le |W|\eta. \quad (2)$$

Finally, comparing the bounds in (1) and (2), we conclude that $\mu \le \eta/((1-\nu)(1-\lambda_0/\tau))$. □

We shall also need a generalization of Lemma 2.3 for the case of weighted graphs. Here we consider bipartite graphs $G = G(L \cup R, E)$ whose edges are assigned weights in the interval $[0, 1]$ satisfying the following property: for every right vertex $y \in R$, all the edges incident on y are assigned the same weight. Let $w_0, w_1, \ldots$ be the distinct weights of the edges of G, in decreasing order. The vertex set R of such a weighted graph is naturally partitioned into subsets $W_0, W_1, \ldots$, where W_i is the subset of all those vertices in R whose incident edges have weight w_i. Intuitively, such a partitioning of R defines a new induced graph G' where a vertex in W_i is present in G' with probability w_i. (In the setting of Lemma 2.3, there are two sets $W_0 = W$ and $W_1 = R \setminus W$ with $w_0 = 1$ and $w_1 = 0$.)

Suppose that the edges of G are partitioned into red and green edges. Let *Red* denote the set of all red edges, and, for every $x \in L$, let Red_x denote the set of all red edges incident to x.

First, consider the experiment where one picks a vertex $y \in R$ with probability proportionate to w_i, where $y \in W_i$, and then picks a uniformly random edge incident to y. What is the probability of picking a red edge?

Let $wt : E \to [0, 1]$ be the edge weight function for our graph $G = G(L \cup R, E)$, and let D be the right degree of the graph G. For a fixed red edge e of G, the probability of choosing this edge in the random experiment described above is

$$\frac{wt(e)}{\sum_{i\geq 0} |W_i| w_i} \cdot \frac{1}{D},$$

where $wt(e)/(\sum_{i\geq 0} |W_i| w_i)$ is the probability of choosing the vertex $y \in R$ that is the end vertex of the edge e, and $1/D$ is the probability of picking one of the D edges incident to y. The probability of picking some red edge is then simply the sum of the probabilities of picking an edge e over all red edges e of G.

Next consider the following experiment. Pick a vertex $x \in L$ uniformly at random, then pick an edge e incident to x with probability proportionate to $wt(e)$ (i.e., the probability $wt(e)/(\sum_{e' \in E_x} wt(e'))$). The probability $\xi(x)$ of picking a red edge incident to x is then the sum of the probabilities of choosing an edge e incident to x, over all red edges e incident on x. Finally, the overall probability of picking a red edge in this experiment is simply the average $\mathbf{Exp}_{x\in L}[\xi(x)]$.

The next lemma basically says that, for sampler graphs G, the probabilities of picking a red edge in the two experiments described above are almost the same. More precisely, we have the following:

Lemma 2.4. *Suppose $G = G(L \cup R, E)$ is a λ-sampler with the right degree D. Let $wt : E \to [0, 1]$ be the weight function over the edges of G such that, for each $y \in R$, the weights of the edges $e \in E_y$ incident to y are the same. Let $w_0, w_1, \ldots$ be the distinct weights of the edges of G in decreasing order, and let $W_0, W_1, \ldots$ be the partitioning of the vertex set R so that each W_i is the subset of all those vertices in R whose incident edges have the weight w_i. Suppose that W_0 has the measure at least τ in the set R.*

Let $Col : E \to \{red, green\}$ be any coloring of the edges of G. For each $x \in L$, let Red_x be the set of all red edges incident to x, and let Red be the set of all red edges in G. Suppose that the total weight of red edges $\sum_{e\in Red} wt(e)$ is at most $\eta D|R|$, and let $\xi(x) = (\sum_{e\in Red_x} wt(e))/(\sum_{e\in E_x} wt(e))$.

Then, for all $0 < \nu, \beta < 1$ and $\lambda_0 = \lambda(\beta, \nu)$, we have

$$\mathbf{Exp}_{x\in L}[\xi(x)] \leq \max\left\{ \frac{\eta|R|}{(1-\nu)(1-\lambda_0/\tau)\sum_{i\geq 0}|W_i|w_i}, \beta \right\}.$$

Proof. Let $\mu = \mathbf{Exp}_{x\in L}[\xi(x)]$. If $\mu \leq \beta$, then we are done. So for the rest of the proof, we will suppose that $\mu > \beta$. We will bound the following sum from below and from above:

$$\sum_{i\geq 0} \sum_{y\in W_i} w_i \cdot \mathbf{Exp}_{x\in N(y)}\big[\xi(x)\big]. \tag{3}$$

To bound it from below, let $Bad \subseteq R$ be the subset of those vertices u where $\mathbf{Exp}_{x\in N(u)}[\xi(x)] < (1-\nu)\mu$. Since G is a λ-sampler, we get that Bad has measure at most λ_0 in R. Each vertex y outside the set Bad contributes at least $w_i(1-\nu)\mu$ to the sum (3) for $y \in W_i$. Since $w_0 \geq w_1 \geq \cdots$, the sum of such contributions is minimized

when all the bad vertices are in W_0, i.e., $Bad \subseteq W_0$ (otherwise, we can always make the sum smaller by placing a bad vertex into W_0 and creating a good vertex in some W_i for $i > 0$). Since W_0 has measure at least τ, we get that Bad has measure at most λ_0/τ in W_0, and so

$$\sum_{i\geq 0}\sum_{y\in W_i} w_i \cdot \mathbf{Exp}_{x\in N(y)}\big[\xi(x)\big] \geq (1-\lambda_0/\tau)|W_0|w_0(1-\nu)\mu + \sum_{i\geq 1}|W_i|w_i(1-\nu)\mu$$
$$\geq (1-\lambda_0/\tau)(1-\nu)\mu\sum_{i\geq 0}|W_i|w_i.$$

For the upper bound on the sum in (3), we use the definition of the $\xi(x)$, change the order of summation, and finally use the definition of η to obtain the following:

$$\begin{aligned}\sum_{i\geq 0}\sum_{y\in W_i} w_i \cdot \mathbf{Exp}_{x\in N(y)}\big[\xi(x)\big] &= (1/D)\sum_{y\in R}\sum_{x\in N(y)}\xi(x)wt\big((x,y)\big)\\ &= (1/D)\sum_{x\in L}\xi(x)\sum_{e\in E_x} wt(e)\\ &= (1/D)\sum_{x\in L}\sum_{e\in Red_x} wt(e)\\ &= (1/D)\sum_{e\in Red} wt(e)\\ &\leq |R|\eta.\end{aligned}$$

Comparing the obtained lower and upper bounds, we conclude that

$$\mu \leq \frac{|R|\eta}{(1-\nu)(1-\lambda_0/\tau)\sum_{i\geq 0}|W_i|w_i},$$

completing the proof of the lemma. □

We conclude this section by showing that the direct product gives rise to a sampler. Consider the following bipartite graph $G = G(L \cup R, E)$: the set of left vertices L is the set of n-bit strings $\{0,1\}^n$; the right vertices R are all k-tuples of n-bit strings $\{0,1\}^{nk}$; for every $y = (u_1, \ldots, u_k) \in R$, there are k edges $(y, u_1), \ldots, (y, u_k) \in E$.

Lemma 2.5. *The graph G defined above is a λ-sampler for $\lambda(\mu, \nu) = e^{-\nu^2\mu k/2}$.*

Proof. This is immediate from Lemma 2.1. □

3. Proof of the Main Theorem

The proof is by contradiction. Suppose that a solver $\bar{C}$ solves the direct product puzzle P^k with fewer than $(1-\gamma)\delta k$ mistakes for more than ϵ fraction of k-tuples of puzzle

instances. We will describe a solver $\mathcal{C}$ which solves the puzzle P with probability at least $1 - \delta$, where the probability is over the internal randomness of the solver and uniformly chosen $\alpha \in \{0,1\}^n$.

We first give the proof under the simplifying assumptions that all puzzles are non-ambiguous (i.e., a puzzle x uniquely determines the random tape α that generated x) and that we can test if a given k-tuple of puzzle instances $(x_1, \ldots, x_k)$ is such that $\bar{C}(x_1, \ldots, x_k)$ makes fewer than $(1 - \gamma)\delta k$ mistakes. Later we remove these assumptions.

3.1. *Proof Under Simplifying Assumptions*

Let *Good* be the subset of k-tuples of puzzle instances where $\bar{C}$ makes few mistakes. More precisely, *Good* is the set of those k-tuples of random tapes $(\alpha_1, \ldots, \alpha_k)$ such that, for the corresponding k-tuple of puzzles $(x_1, \ldots, x_k)$, the solver $\bar{C}$ makes fewer than $(1 - \gamma)\delta k$ mistakes. Since we assume that all puzzles are non-ambiguous, we can define the set *Good*$'$ of all those k-tuples of puzzles $(x_1, \ldots, x_k)$ such that the k-tuple of corresponding random tapes $(\alpha_1, \ldots, \alpha_k)$ is in G. That is, *Good*$'$ is the set of all k-tuples of puzzle instances where the solver $\bar{C}$ makes fewer than $(1 - \gamma)\delta k$ mistakes.[2] Our second assumption is that we have an oracle for testing membership in the set *Good*$'$.

Consider the following algorithm $\mathcal{C}$:

> On input x, choose $k - 1$ random tapes $\alpha_1, \ldots, \alpha_{k-1}$ uniformly at random. Let $x_1, \ldots, x_{k-1}$ be the puzzles corresponding to the chosen random tapes. Pick $i \in [k]$ at random and set $\bar{x} = (x_1, \ldots, x_{i-1}, x, x_i, \ldots, x_{k-1})$. Test if $\bar{x} \in$ *Good*$'$ (using the assumed membership oracle for *Good*$'$). If $\bar{x} \in$ *Good*$'$, then output $\bar{C}(\bar{x})_i$; otherwise repeat with new random α's and i. If no output is produced within $4\ln(20/\gamma\delta)/\epsilon$ iterations, then output the error symbol $\perp$.

We want to analyze the success probability of solver $\mathcal{C}$ on a given input x. To this end, we need to argue that (1) the probability of the timeout is small and (2) conditioned on the output being different from $\perp$, it is a correct output with high probability (greater than $1 - \delta$).

Recall the λ-sampler G defined at the end of Sect. 2.2. It has as its left vertices all possible n-bit random tapes α and as its right vertices all possible k-tuples of such tapes. For a left vertex α, its neighbors in G correspond to all possible ways of embedding this α in a k-tuple, as done by our algorithm $\mathcal{C}$. The algorithm $\mathcal{C}$ times out on an input x corresponding to the random tape α iff it never samples a neighbor w of α in G such that $w \in$ *Good*. To bound the probability of timeout, we consider the set $H \subseteq \{0,1\}^n$ of all those left vertices α of G such that α has less than $\epsilon/4$ fraction of its neighbors fall into the set *Good*.

Claim 3.1. *The set H has density at most $\gamma\delta/5$.*

[2] Note that the set *Good*$'$ does not make sense if one allows ambiguous puzzles, as the same instance x may be considered solved correctly or incorrectly by $\bar{C}$ depending on the particular random tape α used to generate that x.

Proof. Suppose that the density of H is greater than $\beta = \gamma\delta/5$. Let $H' \subseteq H$ be any subset of H of density exactly β. By our assumption, we have that $\mathbf{Pr}_{\alpha\in L, w\in N(\alpha)}[\alpha \in H' \,\&\, w \in \mathit{Good}] < \beta\epsilon/4$. On the other hand, by Lemma 2.2 we get that the same probability is at least $\beta(\epsilon - \lambda_0)/3$ for $\lambda_0 = \lambda(\beta, 2/3)$. This is a contradiction since $\lambda_0 \le \epsilon/4$ from the assumption of the theorem. □

Claim 3.2. *For every $\alpha \notin H$ and the puzzle x corresponding to that random tape α, we have $\mathbf{Pr}[\mathcal{C}(x) = \perp] \le \gamma\delta/20$, where the probability is over the internal randomness of $\mathcal{C}$.*

Proof. By the definition of H, we get that the probability of timeout on any given $\alpha \notin H$ is at most $(1 - \epsilon/4)^{4\ln(20/\gamma\delta)/\epsilon} \le \gamma\delta/20$. □

Next we bound the probability of $\mathcal{C}$ making a mistake, conditioned on $\mathcal{C}$ outputting something other than $\perp$. First we observe that $\mathcal{C}$ produces a definite answer on an input x corresponding to the random tape α exactly when it samples a neighbor w of α in the graph G such that $w \in \mathit{Good}$. Consider the subgraph G' of G induced by removing all right vertices of G except those in Good. For each edge in G' between $w = (\alpha_1, \ldots, \alpha_k) \in R$ and $\alpha_i \in L$, color this edge red if $\bar{C}(x_1, \ldots, x_k)_i$ is wrong and color it green otherwise. Then the requisite conditional probability of $\mathcal{C}$ making a mistake is exactly $\mathbf{Pr}_{\alpha\in L, w\in N_{G'}(\alpha)}[(\alpha, w) \text{ is red}]$.

Claim 3.3. $\mathbf{Pr}_{\alpha\in\{0,1\}^n}[\mathcal{C}(x) \textit{ is wrong} \mid \mathcal{C}(x) \neq \perp] \le \delta - \gamma\delta/4$, *where x is the puzzle corresponding to the random tape α.*

Proof. By the discussion above, the required conditional probability is exactly $\mathbf{Pr}_{\alpha\in L, w\in N_{G'}(\alpha)}[(\alpha, w) \text{ is red}]$. Observe that, by the definition of the set Good, the number of red edges in the graph G' is at most $(1 - \gamma)\delta k|\mathit{Good}|$. By Lemma 2.3 applied to G' with $\eta = (1 - \gamma)\delta$, $\beta = \delta/2$, and $\nu = \gamma/2$, we get that this probability is at most $\max\{(1 - \gamma)\delta/((1 - \gamma/2)(1 - \lambda_0/\epsilon)), \delta/2\}$, where $\lambda_0 = \lambda(\delta/2, \gamma/2)$. This is at most $\delta - \gamma\delta/4$ if $\lambda_0/\epsilon < \gamma/4$. □

Finally, we have

$$\mathbf{Pr}_{\alpha\in\{0,1\}^n}\big[\mathcal{C}(x) \text{ is wrong}\big] = \frac{1}{2^n}\sum_{\alpha\in H}\mathbf{Pr}\big[\mathcal{C}(x) \text{ is wrong}\big] + \frac{1}{2^n}\sum_{\alpha\notin H}\mathbf{Pr}\big[\mathcal{C}(x) \text{ is wrong}\big]. \tag{4}$$

The first term on the right-hand side of (4) is at most $\gamma\delta/5$ by Claim 3.1. For the second term, we upperbound $\mathbf{Pr}[\mathcal{C}(x) \text{ is wrong}]$ by $\mathbf{Pr}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp] + \mathbf{Pr}[\mathcal{C}(x) = \perp]$. We know by Claim 3.2 that, for each $\alpha \notin H$, $\mathbf{Pr}[\mathcal{C}(x) = \perp] \le \gamma\delta/20$. Thus we get that $\mathbf{Pr}_{\alpha\in\{0,1\}^n}[\mathcal{C}(x) \text{ is wrong}]$ is at most

$$\gamma\delta/5 + \gamma\delta/20 + \frac{1}{2^n}\sum_{\alpha\notin H}\mathbf{Pr}\big[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp\big]$$

$$\le \gamma\delta/4 + \mathbf{Pr}_{\alpha\in\{0,1\}^n}\big[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp\big],$$

which is at most δ by Claim 3.3.

3.2. *Proof without Simplifying Assumptions*

Here we explain how to prove our main theorem without any simplifying assumptions. Since we cannot test membership in the set *Good* of k-tuples where $\bar{C}$ makes fewer than $(1-\gamma)\delta k$ errors, we will make a "soft" (probabilistic) decision of how likely a given k-tuple $\bar{x}$ is in *Good* based on the number of correct answers of $\bar{C}(\bar{x})$ in those $(k-1)$ positions where we know the α's (since we have generated them ourselves). The fewer errors we see, the more likely we are to believe the answer of $\bar{C}(\bar{x})$ for the position where the real input x was placed.

More precisely, we will use the following subroutine TRS (Trust Reducing Strategy):

> On inputs $\bar{x} = (x_1, \ldots, x_k)$, $i \in [k]$, and α_j's corresponding to the x_j's for $j \in [k] \setminus \{i\}$, compute the number *err* of errors made by $\bar{C}(\bar{x})$ in positions other than i, that is, $err = |\{j \in [k] \setminus \{i\} \mid \neg R((x_j, \alpha_j), \bar{C}(\bar{x})_j)\}|$. Set $\Delta = err - (1-\gamma)\delta k$. If $\Delta \leq 0$, then output $\bar{C}(\bar{x})_i$ with probability 1. Otherwise, for the parameter $\rho = 1 - \gamma/10$, output $\bar{C}(\bar{x})_i$ with probability ρ^{Δ} and output $\perp$ with probability $1 - \rho^{\Delta}$.

Now our new randomized algorithm $\mathcal{C}$ is as follows:

> On input x, choose $(k-1)$ random tapes $\alpha_1, \ldots, \alpha_{k-1} \in \{0,1\}^n$ uniformly at random. Let $x_1, \ldots, x_{k-1}$ be the puzzles corresponding to the chosen random tapes. Pick $i \in [k]$ at random and set $\bar{x} = (x_1, \ldots, x_{i-1}, x, x_i, \ldots, x_{k-1})$. Run the subroutine TRS on the inputs $\bar{x}$, i, and $\alpha_1, \ldots, \alpha_{k-1}$. If TRS returns a value $y \neq \perp$, then output y; otherwise repeat with new random α's and i. If no output is produced within $4\ln(20/\gamma\delta)/\epsilon$ iterations, then output $\perp$.

We will analyze this algorithm $\mathcal{C}$ using the λ-sampler G defined at the end of Sect. 2.2. Recall that G has as its left vertices all possible n-bit random tapes α and as its right vertices all possible k-tuples of such tapes. For a left vertex α, its neighbors in G correspond to all possible ways of embedding this α in a k-tuple.

First we will bound the probability of timeout of $\mathcal{C}$. Let x be an input corresponding to the random tape α. The k-tuple $\bar{x} = (x_1, \ldots, x_{i-1}, x, x_i, \ldots, x_{k-1})$ of puzzles constructed by the algorithm $\mathcal{C}$ corresponds to the k-tuple $\bar{\alpha} = (\alpha_1, \ldots, \alpha_{i-1}, \alpha, \alpha_i, \ldots, \alpha_{k-1})$ of random tapes. If $\bar{\alpha} \in \mathit{Good}$, then the TRS subroutine will return $\bar{C}(\bar{x})_i \neq \perp$ with probability 1. Hence, the probability of timeout on x is at most the probability that $\mathcal{C}$ never samples a neighbor $\bar{\alpha} \in \mathit{Good}$ of α in the graph G. As in the previous subsection, we consider the set $H \subseteq \{0,1\}^n$ of all those left vertices α of G such that α has less than $\epsilon/4$ fraction of its neighbors fall into the set *Good*. We get the following analogs of Claims 3.1 and 3.2 with exactly the same proofs.

Claim 3.4. *The set H has density at most $\gamma\delta/5$.*

Claim 3.5. *For every $\alpha \notin H$ and the puzzle x corresponding to that random tape α, we have* $\mathbf{Pr}[\mathcal{C}(x) = \perp] \leq \gamma\delta/20$, *where the probability is over the internal randomness of $\mathcal{C}$.*

Next we analyze the probability of $\mathcal{C}$ outputting a wrong answer, conditioned on its output being something other than $\perp$. For each edge $((\alpha_1, \ldots, \alpha_k), \alpha_i)$ of the graph G, we color this edge green if $\bar{C}(x_1, \ldots, x_k)_i$ is correct, and we color it red otherwise.

Consider the following random experiment $\mathcal{E}$:

> Pick a random $\alpha \in L$ and its random incident edge $e = (\alpha, \bar{\alpha})$ in G for $\bar{\alpha}$ containing α in position $i \in [k]$. Let err be the number of errors made by $\bar{C}(\bar{x})$ in positions other than i, and let $\Delta = err - (1-\gamma)\delta k$. If $\Delta \leq 0$, output the edge e. Otherwise, output e with probability ρ^{Δ} (for $\rho = (1-\gamma/10)$) and output $\perp$ with probability $1 - \rho^{\Delta}$.

For each $\alpha \in \{0,1\}^n$ and the puzzle x corresponding to the random tape α, we have

$$\begin{aligned}&\mathbf{Pr}\big[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp\big] \\ &\quad = \mathbf{Pr}[\mathcal{E} \text{ outputs red edge incident to } \alpha \mid \mathcal{E} \text{ outputs some edge incident to } \alpha], \quad (5)\end{aligned}$$

where the first probability is over internal randomness of $\mathcal{C}$, and the second probability is over the random choices of $\mathcal{E}$ for the fixed α (i.e., over the random choice of an edge e incident to α, and the random choice whether e is output).

Rather than analyzing the experiment $\mathcal{E}$, however, we will consider another experiment that is the same as $\mathcal{E}$ except that err is defined as the total number of errors made by $\bar{C}(\bar{x})$ in *all* positions (i.e., including the position i). That is, we consider the following experiment $\mathcal{E}'$:

> Pick a random $\alpha \in L$ and its random incident edge $e = (\alpha, \bar{\alpha})$ in G for $\bar{\alpha}$ containing α in position $i \in [k]$. Let err be the number of errors made by $\bar{C}(\bar{x})$ in all positions, and let $\Delta = err - (1-\gamma)\delta k$. If $\Delta \leq 0$, output the edge e. Otherwise, output e with probability ρ^{Δ} (for $\rho = (1-\gamma/10)$) and output $\perp$ with probability $1 - \rho^{\Delta}$.

Claim 3.6. *For each edge e of G, we have*

$$\mathbf{Pr}[\mathcal{E}' \textit{ outputs } e] \leq \mathbf{Pr}[\mathcal{E} \textit{ outputs } e] \leq (1/\rho)\mathbf{Pr}[\mathcal{E}' \textit{ outputs } e].$$

Proof. Let $p = \mathbf{Pr}[\mathcal{E} \text{ outputs } e]$, and let $p' = \mathbf{Pr}[\mathcal{E}' \text{ outputs } e]$. If edge e is green or $\Delta \leq 0$, then $p' = p$. If e is red and $\Delta > 0$, then $p' = \rho p$. In either case, we have $p' \leq p$ and $p \leq (1/\rho)p'$, as required. □

As a corollary of Claim 3.6, we get the following:

Claim 3.7. *For each $\alpha \in \{0,1\}^n$ and the puzzle x corresponding to α, we have that*

$$\begin{aligned}&\mathbf{Pr}[\mathcal{E} \textit{ outputs red edge incident to } \alpha \mid \mathcal{E} \textit{ outputs some edge incident to } \alpha] \\ &\quad \leq (1/\rho) \cdot \mathbf{Pr}[\mathcal{E}' \textit{ outputs red edge incident to } \alpha \mid \mathcal{E}' \textit{ outputs some edge incident to } \alpha].\end{aligned}$$

Proof. The proof is by Claim 3.6 and the definition of conditional probability. □

Now we can prove the following analog of Claim 3.3.

Claim 3.8. $\mathbf{Pr}_{\alpha \in \{0,1\}^n}[\mathcal{C}(x) \textit{ is wrong} \mid \mathcal{C}(x) \neq \perp] \leq \delta - \gamma\delta/4$.

Proof. By (5) and Claim 3.7, we get that $\mathbf{Pr}_{\alpha\in\{0,1\}^n}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp]$ is at most

$$(1/\rho)\cdot \mathbf{Exp}_{\alpha\in L}\big[\mathbf{Pr}[\mathcal{E}' \text{ outputs red edge incident to } \alpha \mid \mathcal{E}' \text{ outputs some edge incident to } \alpha]\big]. \tag{6}$$

To upperbound the conditional probability of getting a red edge in the experiment $\mathcal{E}'$, we assign weights to the edges of our graph $G = G(L \cup R, E)$ as follows: An edge $(\alpha, \bar{\alpha}) \in E$ between $\alpha \in L$ and $\bar{\alpha} \in R$ gets the weight $wt(e) = \rho^{\Delta}$, where Δ is as in the definition of the experiment $\mathcal{E}'$ (i.e., Δ is the total number of errors of $\bar{C}(\bar{x})$ minus $(1-\gamma)\delta k$).

For each $\alpha \in L$, let Red_α denote the set of all red edges incident to α, and let

$$\xi(\alpha) = \frac{\sum_{e\in Red_\alpha} wt(e)}{\sum_{e\in E_\alpha} wt(e)},$$

where E_α denotes the set of all edges incident to α. The expectation in (6) is exactly $\mu = \mathbf{Exp}_{\alpha\in L}[\xi(\alpha)]$.

Let Red be the set of all red edges in G, and let $\eta = (1/k|R|)\sum_{e\in Red} wt(e)$. Let us partition the set $(\{0,1\}^n)^k$ of k-tuples into the subsets $Good_i$ for $i \geq 0$, where $Good_0 = Good$, and for each $i \geq 1$, $Good_i$ contains all those k-tuples $\bar{\alpha} \in R$ where $\bar{C}(\bar{x})$ makes exactly $((1-\gamma)\delta k + i)$ errors.

Apply Lemma 2.4 to G, the partitioning $R = \bigcup_{i\geq 0} Good_i$ with the corresponding weights $\rho^0, \rho^1, \ldots$, and the measure τ of $Good_0$ being at least ϵ. For a parameter β (to be determined later) and $\lambda_0 = \lambda(\beta, \nu)$, we get that μ is at most the maximum of β and the following expression:

$$\frac{|R|\eta}{(1-\nu)(1-\lambda_0/\epsilon)\sum_{i\geq 0}|Good_i|\rho^i}. \tag{7}$$

By the definition of the sets $Good_i$, we get that $\eta \leq (1/k|R|)\sum_{i\geq 0}|Good_i|((1-\gamma)\delta k + i)\rho^i$. Using this bound on η, we can upperbound the expression in (7) by

$$\frac{\sum_{i\geq 0}|Good_i|((1-\gamma)\delta + (i/k))\rho^i}{(1-\nu)(1-\lambda_0/\epsilon)\sum_{i\geq 0}|Good_i|\rho^i}. \tag{8}$$

For $t = \gamma\delta k/4$, let us split the sum in the numerator of (8) into two sums: for $0 \leq i \leq t$ and for $i > t$. We can bound each of these two sums as follows:

$$\sum_{i=0}^{t}|Good_i|\big((1-\gamma)\delta + (i/k)\big)\rho^i \leq \big((1-\gamma)\delta + t/k\big)\sum_{i=0}^{t}|Good_i|\rho^i$$
$$\leq (1-3\gamma/4)\delta\sum_{i\geq 0}|Good_i|\rho^i$$

and

$$\sum_{i>t}|Good_i|\big((1-\gamma)\delta + (i/k)\big)\rho^i \leq \rho^t|R|.$$

Plugging these bounds into (8) and recalling that $|Good_0| \geq \epsilon|R|$, we upperbound (8) by

$$\frac{1}{(1-\nu)(1-\lambda_0/\epsilon)}\left((1-3\gamma/4)\delta + \frac{\rho^t|R|}{\sum_{i\geq 0}|Good_i|\rho^i}\right) \leq \frac{(1-3\gamma/4)\delta + \rho^t/\epsilon}{(1-\nu)(1-\lambda_0/\epsilon)}.$$

Finally, by (6), we get

$$\mathbf{Pr}_{\alpha\in\{0,1\}^n}\left[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp\right] \leq \max\left\{\frac{(1-3\gamma/4)\delta + \rho^t/\epsilon}{\rho(1-\nu)(1-\lambda_0/\epsilon)}, \frac{\beta}{\rho}\right\},$$

which is at most $\delta - \gamma\delta/4$ for $\rho = 1-\gamma/10$, $\beta = (27/40)\delta$, $\nu = (3/10)\gamma$, $\lambda_0/\epsilon \leq \gamma/20$, and $\rho^t/\epsilon \leq \gamma\delta/100$. □

Now we finish the proof of our main theorem.

Proof of Theorem 1.1. The proof follows from Claims 3.4, 3.5, and 3.8 in exactly the same way as the proof of the main theorem under the simplifying assumptions given at the end of Sect. 3.1. □

Remark 3.9. Note that in the proof of the main theorem, we implicitly assumed that the solver $\bar{C}$ is deterministic. The proof can be generalized for a randomized solver $\bar{C}$ by trying to fix the randomness of $\bar{C}$ and evaluating the constructed circuit $\mathcal{C}$ by sampling.

4. Open Problems

While the results here are fairly general, there are some obvious possible extensions. First, can similar results be proved for other domains, such as public-coin protocols [9]. Also, our bounds on the adversary's success probability, although asymptotically exponentially small, are quite weak when applied to concrete problems such as actual CAPTCHA protocols with reasonable numbers of repetitions. Can the bounds be improved quantitatively (getting smaller constant in the exponent for ϵ in Theorem 1.1)? Finally, we would like to find more applications of our results, for example, to such problems as making strong secret agreement protocols from weak ones [4].

References

[1] M. Bellare, R. Impagliazzo, M. Naor, Does parallel repetition lower the error in computationally sound protocols? In *Proceedings of the Thirty-Eighth Annual IEEE Symposium on Foundations of Computer Science* (1997), pp. 374–383

[2] R. Canetti, S. Halevi, M. Steiner, Hardness amplification of weakly verifiable puzzles, in *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005*, pp. 17–33

[3] O. Goldreich, N. Nisan, A. Wigderson, On Yao's XOR-Lemma. Electronic colloquium on computational complexity, TR95-050 (1995)

[4] T. Holenstein, Key agreement from weak bit agreement, in *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing* (2005), pp. 664–673

[5] R. Impagliazzo, R. Jaiswal, V. Kabanets, Approximately list-decoding direct product codes and uniform hardness amplification, in *Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science* (2006), pp. 187–196

[6] R. Impagliazzo, R. Jaiswal, V. Kabanets, Chernoff-type direct product theorems, in *Advances in Cryptology—CRYPTO 2007, Twenty-Seventh Annual International Cryptology Conference* (2007), pp. 500–516

[7] R. Impagliazzo, R. Jaiswal, V. Kabanets, A. Wigderson, Uniform direct-product theorems: Simplified, optimized, and derandomized, in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing* (2008), pp. 579–588

[8] R. Impagliazzo, A. Wigderson, P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma, in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing* (1997), pp. 220–229

[9] R. Pass, M. Venkitasubramaniam, An efficient parallel repetition theorem for Arthur–Merlin games, in *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing* (2007), pp. 420–429

[10] K. Pietrzak, D. Wikstrom, Parallel repetition of computationally sound protocols revisited, in *Theory of Cryptography, Fourth Theory of Cryptography Conference, TCC 2007*, pp. 86–102

[11] L. von Ahn, M. Blum, N.J. Hopper, J. Langford, CAPTCHA: Using hard AI problems for security, in *Advances in Cryptology—EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques* (2003), pp. 294–311

[12] A.C. Yao, Theory and applications of trapdoor functions, in *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science* (1982), pp. 80–91