Chernoff-type Direct Product Theorems

Russell Impagliazzo[1] *, Ragesh Jaiswal[1] **, and Valentine Kabanets[2]

[1] University of California San Diego, USA.
{russell, rjaiswal}@cs.ucsd.edu
[2] Simon Fraser University, Canada.
kabanets@cs.sfu.ca

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 would be an argument system without perfect completeness. A natural approach to boost the gap between legitimate users and attackers would be 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.

1 Introduction

Cryptographic protocols use gaps between the informational and computational abilities of legitimate users and attackers to distinguish the two. Thus, the greater the gap between the ability of legitimate users to solve a type of problem and that of attackers, the more useful the problem is. Ideally, a problem should be reliably easy for legitimate users (in that the chance of failure for legitimate users should be negligible) but reliably hard for attackers (in that the chance of the attacker's success is negligible).

Direct product theorems point out ways to make 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. Examples of such theorems in cryptography include Yao's theorem that weak one-way functions imply strong one-way functions ([23]) and results of [4, 6], showing similar drops even when an attacker cannot know for certain

* Research partially supported by NSF Awards CCR-0313241 and CCR-0515332. Views expressed are not endorsed by the NSF.

** Research partially supported by NSF Awards CCR-0313241, CCR-0515332, CCF-0634909 and CNS-0524765. Views expressed are not endorsed by the NSF.

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 non-trivial. One reason for this is that there are other circumstances where the intuition is incorrect, and many instances are not proportionally harder. Examples of circumstances where direct products fail are parallel repetition for multiple round protocols and for non-verifiable puzzles ([4, 6, 19]).

While standard direct product theorems are powerful, they can only be used to amplify the gap between legitimate users and attackers if legitimate users are successful with high probability. The legitimate user's chance of solving k independent challenges also drops exponentially, so unless the probability of failure isn't much more than $1/k$ to start, both legitimate users and attackers will almost certainly fail to solve all of the problems.

For example, a CAPTCHA protocol is meant to distinguish between humans and programs, usually using a visual challenge based on distorted text with extraneous lines ([2]). 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. [2] prove that sequential repetition of the protocol increases this gap, and refer to [4] for the "more complicated" case of parallel repetition. Indeed, the results of [4] (and improved by [6]) do apply to parallel repetition of CAPTCHA protocols. However, for the reason above, these results only show that the probability of algorithmic success decreases with repetitions, not that the gap improves.

An obvious, intuitive solution to this problem is to make many independent challenges, but accept if the solver is successful on a larger fraction than expected for an attacker. Here, we prove that, for a large variety of problems, this approach indeed amplifies the gap between legitimate users and attackers. The kind of problems we consider are the *weakly verifiable puzzles* of [6], which include challenge-response protocols such as CAPTCHA as a special case. The puzzles are *weakly verifiable* in the sense that, while the generator of the puzzle can verify a solution, the attacker (who is just given the puzzle, not the way it was generated) cannot necessarily verify whether a proposed solution is acceptable. For P a weakly verifiable puzzle, we denote by $P^{k,T}$ the puzzle that asks k independent challenges from P and accepts if at least $(k-T)$ of the responses are correct solutions to P.

Theorem 1 (Main Theorem). *Let P be a weakly verifiable puzzle so that any solver running in time $t(n)$ has probability at least δ of failure (for sufficiently large n). Let k, $\gamma > 0, T = (1-\gamma)\delta k$, and $\epsilon > 2 \cdot e^{-\frac{\gamma^2\delta^2 k}{64}}$, be given parameters (as functions of n). Then no solver running in time $t'(n) = t(n)poly(\epsilon, 1/n, 1/k)$ time can solve $P^{k,T}$ with probability greater than ϵ, for some polynomial poly, for sufficiently large n.*

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.

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 $P^{k,T}$ with all but exponentially small probability. Thus, the above Chernoff-type direct product theorem indeed shows how to amplify any gap between legitimate users and attackers.

1.1 Weakly Verifiable Puzzles

Our result holds for the notion of weakly verifiable puzzles defined by [6].

A *weakly verifiable puzzle* has two components: First, a distribution ensemble $D = D_1, ..., D_n, ...$ on pairs (x, α), where x is called the puzzle and α the check string. n is called the security parameter. Secondly, 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 [6], the distribution D is restricted to being polynomially-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 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 non-uniform adversaries (since many samples from D are required as advice.)

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

1. Consider 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. Consider 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 has no success. Our result shows that, if the parties agree more probably than the eavesdropper can guess the secret, then 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 always 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 $x_i = f(z_i)$ can produce a string closer in relative Hamming distance to $b(x_1)...b(x_n)$ than the hardness of prediction.
4. In the non-uniform setting, our results apply to any function. If f is a function (possibly non-Boolean, or even multi-valued, as long as it takes on 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 isn't necessarily polynomial-time sampleable, so our results would only apply for non-uniform 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.

1.2 Related Work

The notion of a Direct Product Theorem, in which solving multiple instances of a problem simultaneously is proven harder than a single instance, was introduced by Yao in [23]. 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. [8] contains 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 [12]. Such techniques were also used to show a version of the Direct Product Theorem under a more general cryptographic setting by Bellare, Impagliazzo and Naor in [4]. The idea was to show that the soundness error decreases exponentially with parallel repetition in any 3-round challenge-response protocol. This paper also showed that such error amplification might not be possible for a general (> 3)-round protocol. Pietrzak and Wikstrom in [19] extend this negative result. On the positive side, Canetti, Halevi and Steiner in [6] used ideas from [4] to define a general class of *weakly verifiable puzzles* for which they show parallel repetition amplifies hardness, also giving a quantitative improvement over [4]. More recently, Pass and Venkitasubramaniam [18] show similar positive results for constant round public coin protocols. Note that all the results mentioned above, consider parallel repetition without threshold i.e. they consider the hardness of answering all the instances of the parallel repetition question simultaneously.

In this paper, we use the Sampling Lemma (Lemma 1) from [11] in an essential manner. The proof of this Lemma uses ideas from Raz's parallel repetition paper [20].

1.3 Techniques

Our main lemma shows 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 $(1-\delta)$. This algorithm is a version of the trust-reducing strategies from [12, 4]. In a trust-reducing strategy, the real puzzle is hidden among $(k-1)$ randomly generated puzzles, and the number of mistakes the subroutine makes on the random puzzles is used to compute a probability that the algorithm accepts the answer for the real puzzle.

However, we need to deviate substantially from the analysis in the previous papers. The previous work considered the performance of the strategy on a set H of "hard" instances, and showed that if $|H| \geq \delta 2^n$, then the strategy worked almost certainly on random elements of H. (Thus the fraction of puzzles where the strategy fails with reasonable probability would be at most δ.) In contrast, in the threshold scheme, it suffices for the adversary to answer correctly only on the instances outside a "really hard" set H' of size $(1-\gamma)\delta$, and make an error on the instances in H'. Since H' could be a large $(1-\gamma)$ fraction of H, the conditional probability of success on H of any strategy is at most γ.

In order to get around this obstacle, we need a more global way of analyzing the trust-reducing strategy. Our main tool for doing this is a sampling lemma (from [11]) that can be used to show that the strategy has approximately the same success probability on almost all instances. This allows us us to analyze the strategy on random instances, and infer similar success for almost all instances.

2 Preliminaries

Definition 1. *For any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes sampling an element from the distribution $\mathcal{D}$, and $\mathcal{D}(x)$ denotes the probability of sampling the element x.*

Definition 2. *Given two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over $\{0,1\}^n$, the statistical distance $Dist(\mathcal{D}_1, \mathcal{D}_2)$ between them is defined as*

$$Dist(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2} \sum_{x \in \{0,1\}^n} |\mathbf{Pr}[\mathcal{D}_1(x)] - \mathbf{Pr}[\mathcal{D}_2(x)]|$$

Let $\mathcal{U}$ be the uniform distribution on $\{0,1\}^n$. Consider the following distribution over $\{0,1\}^n$. Pick an m tuple of n-bit string $(x_1, \ldots, x_m)$ uniformly at random and output x_i for a randomly chosen $i \in [m]$. The distribution is equivalent to $\mathcal{U}$ if the tuple is randomly chosen from $\{0,1\}^{nm}$. The next lemma shows that the distribution is close to uniform even when the tuple is chosen randomly from a subset $G \subseteq \{0,1\}^{nm}$ of size $\epsilon 2^{nm}$.

Lemma 1 (Sampling Lemma). *Let $G \subseteq \{0,1\}^{mn}$ be any subset of size $\epsilon 2^{mn}$. Let $\mathcal{U}$ be a uniform distribution on the set $\{0,1\}^n$, and let $\mathcal{D}$ be the distribution defined as follows: pick a tuple $(x_1, \ldots, x_m)$ of n-bit strings uniformly from the set G, pick an index i uniformly from $[m]$, and output x_i. Then the statistical distance between the distributions $\mathcal{U}$ and $\mathcal{D}$ is less than $0.6\sqrt{\frac{\log 1/\epsilon}{m}}$.*

See [11] for the proof of this lemma. The following corollary will be used in the proof of our main result.

Corollary 1. *Let $\mathcal{G}$ be a distribution over $\{0,1\}^{nm}$ (which can be viewed as m-tuples of n-bit strings) such that for any $\bar{x} \in \{0,1\}^{nm}$, $\mathcal{G}(\bar{x}) \leq \frac{1}{\epsilon\ 2^{nm}}$. Let $\mathcal{U}$ be a uniform distribution over $\{0,1\}^n$, and let $\mathcal{D}$ be the distribution defined as follows: pick a tuple $(x_1, \ldots, x_m) \leftarrow \mathcal{G}$, pick an index i uniformly from $[m]$, and output x_i. Then the statistical distance between the distributions $\mathcal{U}$ and $\mathcal{D}$ is less than $0.6\sqrt{\frac{\log 1/\epsilon}{m}}$.*

Proof. We can represent the distribution $\mathcal{G}$ as a convex combination of uniform distributions on subsets of size at least $\epsilon 2^{nm}$. We can then apply the Sampling Lemma to each term in the combination to obtain the corollary. $\square$

3 Proof of the Main Theorem

The proof is by contradiction. Given a solver $\bar{C}$ that solves the weakly verifiable puzzle $P^{k,T}$ with probability at least ϵ, we give a solver $\mathcal{C}$ which solves the puzzle P with probability at least $(1-\delta)$. The probability of success is over the internal randomness of the solver and uniformly chosen $\alpha \in \{0,1\}^n$.

Let G be the subset of $\bar{\alpha} = (\alpha_1, \ldots, \alpha_k) \in (\{0,1\}^n)^k$ where[3]

$$|\{i : \neg R((x_i, \alpha_i), \bar{C}(x_1, \ldots, x_k)_i)\}| \leq (1-\gamma)\delta k$$

So G denotes the "good" subset of $\bar{\alpha}$'s for the solver $\bar{C}$ where we have minimum guarantee of ϵ. In order to illustrate the ideas of the proof, we first prove the Theorem assuming access to an oracle $\mathcal{O}^G$ deciding the membership of a given tuple $(\alpha_1, \ldots, \alpha_k)$ in the "good" set G. We then drop this assumption by, essentially, imitating the behavior of the oracle.

3.1 Assuming Oracle $\mathcal{O}^G$ Exists

In this subsection, in order to illustrate the ideas of the proof in a simplified setting, we temporarily assume that there is an oracle $\mathcal{O}^G$ which tells if a given tuple $(\alpha_1, \ldots, \alpha_k)$ belongs to the "good" set G. This subsection is to develop

[3] For a string α_i, we implicitly denote the puzzle by x_i.

the reader's intuition, and is not strictly required for the proof of the real case. For this reason, we will slur over some calculations. In the rest of the section, we show how to drop this assumption by approximating oracle $\mathcal{O}^G$ in a computable way. The rest of the section is self-contained, so this subsection, while helpful, may be skipped by the reader. Note that this oracle is unrealistic in many ways, one of which is that it's answer depends on α, when in the real case the Solver will only have x, not α.

Consider the randomized Solver $\mathcal{C}$ defined in Figure 1 which is allowed a special output $\perp$ which is considered as an incorrect answer in the analysis.

INPUT: $z = (x, \alpha)$
OUTPUT: y
ORACLE ACCESS: Solver $\bar{C}$ and $\mathcal{O}^G$
PARAMETERS: $\epsilon \geq 2 \cdot e^{-\frac{\gamma^2 \delta^2 k}{64}}$, *timeout* $= \frac{4n}{\epsilon}$.

1. Repeat lines 2-6 for at most *timeout* times:
2. Choose $i \in [k]$ uniformly at random.
3. Choose $\alpha_1, \ldots, \alpha_{k-1} \in \{0,1\}^n$ uniformly at random.
4. Let $\bar{\alpha} \leftarrow (\alpha_1, \ldots, \alpha_{i-1}, \alpha, \alpha_i, \ldots, \alpha_{k-1})$.
5. **If** $\mathcal{O}^G(\bar{\alpha}) = 1$
6. then **output** $y = \bar{C}(\bar{x})_i$, *//where the elements of $\bar{x}$ are the puzzles* *// generated from check strings $\bar{\alpha}$.*
7. **output** $\perp$

Solver 1: Randomized Solver $\mathcal{C}$ given $\bar{C}$ and $\mathcal{O}^G$ as oracle

We want to analyze the success probability of solver $\mathcal{C}$ on a given input $z = (x, \alpha)$. To this end, we need to argue that (1) the probability of the timeout (i.e., of outputting $\perp$ in line 7) is small, and (2) conditioned on the output being different from $\perp$, it is a correct output with high probability (greater than $1-\delta$).

We will focus on analyzing the conditional success probability in (2), i.e., $\mathbf{Pr}_{i,\alpha_1,\ldots,\alpha_{k-1}}[\mathcal{C}(\alpha) \text{ is correct} \mid \text{output} \neq \perp]$, for a given input $z = (x, \alpha)$ to $\mathcal{C}$. Observing that $\mathcal{C}$ outputs something other than $\perp$ exactly when the tuple $\bar{\alpha}$ built in line 4 is in the set G, we can rewrite this conditional probability as $\mathbf{Pr}_{i\in[k],\bar{\alpha}=(\alpha_1,\ldots,\alpha_k)\in G}[\mathcal{C}(\alpha_i) \text{ is correct} \mid \alpha_i = \alpha]$, where i is chosen uniformly from $[k]$, and $\bar{\alpha}$ uniformly from G.

Let $\mathcal{D}(\alpha) = \mathbf{Pr}_{i\in[k],\bar{\alpha}\in G}[\alpha_i = \alpha]$, and let $\mathcal{U}$ be the uniform distribution on αs. Using our Sampling Lemma, we will argue that the distributions $\mathcal{D}$ and $\mathcal{U}$ are statistically close to each other. Using this closeness, we can finish the analysis of the success probability of solver $\mathcal{C}$ as follows. The conditional success probability of $\mathcal{C}$ for a random input α is $\sum_\alpha \mathbf{Pr}_{i,\bar{\alpha}\in G}[\mathcal{C}(\alpha_i) \text{ is correct} \mid \alpha_i = \alpha] * \mathcal{U}(\alpha)$, which is approximately equal to $\sum_\alpha \mathbf{Pr}_{i,\bar{\alpha}\in G}[\mathcal{C}(\alpha_i) \text{ is correct} \mid \alpha_i = \alpha] * \mathcal{D}(\alpha)$. The latter expression is exactly $\mathbf{Pr}_{i,\bar{\alpha}\in G}[\mathcal{C}(\alpha_i) \text{ is correct}]$, which is at least $1-(1-\gamma)\delta = 1-\delta+\gamma\delta$, by the definition of G. We will show that the statistical distance between $\mathcal{D}$ and $\mathcal{U}$ and the probability of the timeout of $\mathcal{C}$ are less than $\gamma\delta$, which would imply that $\mathcal{C}$ succeeds on more than $1 - \delta$ fraction of inputs.

To demonstrate the structure of the analysis in the general case, we recast the arguments above as follows. We introduce a certain random experiment $\mathcal{E}$ (see Figure 1), which corresponds to the inner loop of the algorithm $\mathcal{C}$. We then relate the success probability of $\mathcal{C}$ to that of $\mathcal{E}$.

Fig. 1. Experiment $\mathcal{E}$

Experiment $\mathcal{E}$

$(\alpha_1, \ldots, \alpha_k) \stackrel{\$}{\leftarrow} G$ *// Let $x_1, ..., x_k$ be the corresponding puzzles*

$i \stackrel{\$}{\leftarrow} [k]$

output $(\alpha_i, \bar{C}(x_1, \ldots, x_k)_i)$

We say that experiment $\mathcal{E}$ *succeeds* if it outputs a pair (α, y) such that $R((x, \alpha), y)$. Since for each k-tuple in G, $\bar{C}$ outputs a correct answer for at least $1-(1-\gamma)\delta$ fraction of the elements, the probability of success of this experiment is clearly $\geq 1 - (1 - \gamma)\delta$.

Let $\mathcal{D}$ be the probability distribution on the first elements of outputs of $\mathcal{E}$, i.e., $\mathcal{D}(\alpha)$ is the probability that $\mathcal{E}$ outputs a pair (α, y). Let R_α represent the probability that it outputs such a pair with $R((x, \alpha), y)$, and W_α the probability that it outputs such a pair with $\neg R((x, \alpha), y)$. So, $\mathcal{D}(\alpha) = R_\alpha + W_\alpha$. Clearly, we have that $1 - (1 - \gamma)\delta \leq \mathbf{Pr}[\mathcal{E} \text{ succeeds}] = \sum_{\alpha \in \{0,1\}^n} R_\alpha$.

Since $\mathcal{D}$ is sampled by picking a random element of a set G of tuples of size at least $\epsilon 2^{nk}$, from the sampling lemma it is within $0.6\sqrt{\log(1/\epsilon)/k} \leq \gamma\delta/8$ statistical distance of the uniform distribution. In particular, for $H = \{\alpha | \mathcal{D}(\alpha) \leq (1/2)2^{-n}\}$, $|H| \leq (\gamma\delta/4)2^n$.

Let p_α be the probability that a random $\bar{\alpha}$ containing α is in G. Then the expectation of p_α for random α is at least ϵ, and $\mathcal{D}(\alpha) = p_\alpha / \sum_{\alpha'} p_{\alpha'} = 2^{-n}(p_\alpha / \mathbf{Exp}[p_{\alpha'}])$. So all elements not in H have $p_\alpha \geq \epsilon/2$. For each such element, the probability that we get a timeout in $\mathcal{C}$ is at most $(1-p_\alpha)^{timeout} \leq e^{-n}$.

Given that $\mathcal{C}$ on α does not time out, the probability of it succeeding is $R_\alpha / \mathcal{D}(\alpha)$. Thus, the overall probability of success is at least $(\sum_\alpha \mathcal{U}(\alpha) R_\alpha / \mathcal{D}(\alpha)) - \mathbf{Pr}[\mathcal{C} \text{ times out}]$. We get $\sum_\alpha \mathcal{U}(\alpha) R_\alpha / \mathcal{D}(\alpha) = \sum_\alpha (\mathcal{U}(\alpha) - \mathcal{D}(\alpha)) R_\alpha / \mathcal{D}(\alpha) + \sum_\alpha R_\alpha \geq \sum_\alpha (-1)|\mathcal{U}(\alpha) - \mathcal{D}(\alpha)| + (1 - (1 - \gamma)\delta) \geq 1 - (1 - \gamma)\delta - Dist(\mathcal{D}, \mathcal{U}) \geq 1 - (1 - 3/4\gamma)\delta$.

The probability of time-out can be bounded by the probability that $\alpha \in H$ plus the probability of time-out given that $\alpha \notin H$. As previously mentioned, this is at most $\delta\gamma/4 + e^{-n}$, giving a total success probability at least $1 - (1 - \gamma/2)\delta - e^{-n} > 1 - \delta$, as desired.

3.2 Removing the Oracle $\mathcal{O}^G$

We will use the same set of ideas as in the previous subsection while removing the dependency on the simplifying assumptions. The most important assumption made was the existence of the oracle $\mathcal{O}^G$ which helped us to determine if a tuple $(\alpha_1, \ldots, \alpha_k) \in G$. The second assumption that we made was that the randomized solver $\mathcal{C}$ receives as input the pair $z = (x, \alpha)$ which is not a valid assumption since α is supposed to be hidden from the solver.

We get around these assumptions, in some sense, by imitating the combined behavior of the randomized solver and the oracle $\mathcal{O}^G$ of the previous subsection. Below we define a new randomized solver which is only given x as an input, with α being hidden from the solver.

INPUT: x *//corresponding to* α
OUTPUT: y
ORACLE ACCESS: Solver $\bar{C}$
PARAMETERS: $\epsilon \geq 2 \cdot e^{-\frac{\gamma^2 \delta^2 k}{64}}$, *timeout* $= \frac{4n}{\epsilon}$, $t_0 = (1-\gamma)\delta k$, $\rho = 1 - \frac{\gamma\delta}{16}$.

1. Repeat lines 2-10 for at most *timeout* times:
2. *// Subroutine TRS (Trust Reducing Strategy)*
3. Choose $i \in [k]$ uniformly at random.
4. Choose $\alpha_1, \ldots, \alpha_{k-1} \in \{0,1\}^n$ uniformly at random.
5. Let $\bar{x} \leftarrow (x_1, \ldots, x_{i-1}, x, x_i, \ldots, x_{k-1})$.
6. Let $l = \{j : \neg R((x_j, \alpha_j), \bar{C}(x_1, \ldots, x_{k-1})_j), j \neq i\}$
7. **If** $|l| > t_0$
8.   **output** $\bar{C}(x_1, \ldots, x_{k-1})_i$ with probability $\rho^{|l|-t_0}$
9. **else**
10.   **output** $\bar{C}(x_1, \ldots, x_{k-1})_i$ with probability 1
11. **output** $\perp$

Solver 2: Randomized Solver $\mathcal{C}$ given $\bar{C}$ as oracle

To be able to analyze the above solver we abstract out a single execution of the loop $2 - 10$ (the subroutine TRS) and design an experiment $\mathcal{E}_3$ which has similar behavior. To further simplify the analysis we design a simpler experiment $\mathcal{E}_2$ such that (1) analyzing $\mathcal{E}_2$ is easy and (2) $\mathcal{E}_2$ is not too much different from $\mathcal{E}_3$ so that we can easily draw comparisons between them. The description of Experiments $\mathcal{E}_2$ and $\mathcal{E}_3$ is given in Figure 2.

Definition 3. *Experiments $\mathcal{E}_2$ and $\mathcal{E}_3$ are said to* succeed *if they output a correct pair (i.e. a pair (α, y) such that $R((x, \alpha), y)$). The success probability is defined as the probability that a correct pair is produced conditioned on the experiment producing a pair.*

Fig. 2. Experiments $\mathcal{E}_2$ and $\mathcal{E}_3$.

Experiment $\mathcal{E}_2$	Experiment $\mathcal{E}_3$
$(\alpha_1,\ldots,\alpha_k) \xleftarrow{\$} (\{0,1\}^n)^k$	$(\alpha_1,\ldots,\alpha_k) \xleftarrow{\$} (\{0,1\}^n)^k$
$i \xleftarrow{\$} [k]$	$i \xleftarrow{\$} [k]$
$J \leftarrow \{j \mid \neg R((x_j,\alpha_j),\bar{C}(x_1,\ldots,x_k))_j\}$	$J \leftarrow \{j \mid \neg R((x_j,\alpha_j),\bar{C}(x_1,\ldots,x_k))_j\}$
if $\lvert J\rvert > (1-\gamma)\delta k$	**if** $\lvert J\rvert > (1-\gamma)\delta k$
$\quad t = \lvert J\rvert - (1-\gamma)\delta k$	$\quad t = \lvert J\rvert - (1-\gamma)\delta k$
else	**else**
$\quad t = 0$	$\quad$ **output** $(\alpha_i, \bar{C}(x_1,\ldots,x_k)_i)$
output $(\alpha_i, \bar{C}(x_1,\ldots,x_k)_i)$	$\qquad$ with probability 1
$\quad$ with probability ρ^t and $\perp$	**if** $i \in J$
$\quad$ with probability $(1-\rho^t)$	$\quad$ **output** $(\alpha_i, \bar{C}(x_1,\ldots,x_k)_i)$
	$\qquad$ with probability ρ^{t-1} and $\perp$
	$\qquad$ with probability $(1-\rho^{t-1})$
	else
	$\quad$ **output** $(\alpha_i, \bar{C}(x_1,\ldots,x_k)_i)$
	$\qquad$ with probability ρ^t and $\perp$
	$\qquad$ with probability $(1-\rho^t)$

Proof outline. We observe that the success probability of $\mathcal{C}$ on a given input x corresponding to a hidden string α is exactly the success probability of experiment $\mathcal{E}_3$ *conditioned* on the event that $\mathcal{E}_3$ produces a pair $(\alpha,\cdot)$. For a random input x corresponding to a uniformly random string α, the success probability of $\mathcal{C}$ is then $\sum_\alpha \mathbf{Pr}[\mathcal{E}_3 \text{ succeeds} \mid \mathcal{E}_3 \text{ outputs } (\alpha,\cdot)] * \mathcal{U}(\alpha)$, where $\mathcal{U}$ denotes the uniform distribution. On the other hand, the success probability of $\mathcal{E}_3$ can be written as $\sum_\alpha \mathbf{Pr}[\mathcal{E}_3 \text{ succeeds} \mid \mathcal{E}_3 \text{ outputs } (\alpha,\cdot)] * \mathcal{D}_3(\alpha)$, where $\mathcal{D}_3(\alpha)$ is the probability that experiment $\mathcal{E}_3$ produces a pair $(\alpha,\cdot)$ conditioned on $\mathcal{E}_3$ producing some pair (i.e., conditioned on the output of $\mathcal{E}_3$ being different from $\perp$). We then argue that the distributions $\mathcal{U}$ and $\mathcal{D}_3$ are statistically close, and hence the success probability of $\mathcal{C}$ can be lowerbounded with that of $\mathcal{E}_3$. Finally, we lowerbound the success probability of $\mathcal{E}_3$, getting the result for $\mathcal{C}$.

In reality, the success probability of experiment $\mathcal{E}_2$ is easier to analyze than $\mathcal{E}_3$. So we actually show that the conditional success probability of $\mathcal{E}_3$ can be lowerbounded by that of $\mathcal{E}_2$, and then argue that $\mathcal{U}$ is statistically close to $\mathcal{D}_2$, where $\mathcal{D}_2$ is defined for $\mathcal{E}_2$ in the same way as $\mathcal{D}_3$ was defined for $\mathcal{E}_3$ above.

Next we give the details of the proof. We start by analyzing $\mathcal{E}_2$.

Analyzing $\mathcal{E}_2$ Let us partition the k-tuples $(\{0,1\}^n)^k$ into the following subsets:

$$G_0 = G = \{(\alpha_1, ..., \alpha_k) \in \{0,1\}^{nk} : |\{j : \neg R((x_j, \alpha_j), \bar{C}(x_1, ..., x_k)_j)\}| \le (1-\gamma)\delta k\}$$
$$G_1 = \{(\alpha_1, ..., \alpha_k) \in \{0,1\}^{nk} : |\{j : \neg R((x_j, \alpha_j), \bar{C}(x_1, ..., x_k)_j)\}| = (1-\gamma)\delta k + 1\}$$
$$\vdots$$
$$G_{k(1-(1-\gamma)\delta)} = \{(\alpha_1, ..., \alpha_k) \in \{0,1\}^{nk} : |\{j : \neg R((x_j, \alpha_j), \bar{C}(x_1, ..., x_k)_j)\}| = k\}$$

Definition 4. *We let $S_{\not\perp}$ denote the general event that the experiment produces a pair (i.e. does* not *produce $\perp$) and S_c denote the event that the experiment produces a correct output.*

Claim 2 $\mathbf{Pr}[\mathcal{E}_2$ *succeeds*$] \ge \left(1 - \frac{t_0 + \gamma\delta k/2}{k}\right)\left(1 - \frac{\rho^{\gamma\delta k/2}}{\epsilon}\right)$, *where $t_0 = (1-\gamma)\delta k$.*

Proof. Let $t_0 = (1-\gamma)\delta k$ and $\Delta = \gamma\delta k$. Let $\bar{A}$ denote the random tuple chosen in the first step of experiment $\mathcal{E}_2$. Recalling that the success probability of $\mathcal{E}_2$ is defined as the probability of producing a correct pair conditioned on producing a pair as output, we get

$$\begin{aligned}
\mathbf{Pr}[\mathcal{E}_2 \text{ succeeds}] &= \mathbf{Pr}[S_c | S_{\not\perp}] \\
&= \mathbf{Pr}[S_c, S_{\not\perp}] / \mathbf{Pr}[S_{\not\perp}] \\
&= \sum_{\bar{\alpha} \in \{0,1\}^{nk}} \mathbf{Pr}[S_c, S_{\not\perp}, \bar{A} = \bar{\alpha}] / \mathbf{Pr}[S_{\not\perp}] \\
&= \sum_{\bar{\alpha} \in \{0,1\}^{nk}} \mathbf{Pr}[S_c | S_{\not\perp}, \bar{A} = \bar{\alpha}] \cdot \mathbf{Pr}[S_{\not\perp}, \bar{A} = \bar{\alpha}] / \mathbf{Pr}[S_{\not\perp}].
\end{aligned}$$

We will split the set of $\bar{\alpha}$'s into the following three sets:

$$G = G_0, \qquad I = G_1 \cup \ldots \cup G_{\Delta/2}, \qquad B = \{0,1\}^{nk} - G - I,$$

which stand for "good", "intermediate" and "bad", respectively. We note that $\mathcal{E}_2$ performs *well* on tuples in the good subset, *reasonably well* on tuples in the intermediate subset and *poorly* on the tuples in the bad subset. The intuitive idea is that we counter the poor effect of the bad subset of tuples by exponentially weighing down their contribution in the overall probability of success of $\mathcal{E}_2$.

We have

$$\begin{aligned}
\mathbf{Pr}[\mathcal{E}_2 \text{ succeeds}] &\ge \sum_{\bar{\alpha} \in G \cup I} \mathbf{Pr}[S_c | S_{\not\perp}, \bar{A} = \bar{\alpha}] \cdot \mathbf{Pr}[S_{\not\perp}, \bar{A} = \bar{\alpha}] / \mathbf{Pr}[S_{\not\perp}] \\
&\ge \sum_{\bar{\alpha} \in G \cup I} \left(1 - \frac{t_0 + \Delta/2}{k}\right) \cdot \mathbf{Pr}[S_{\not\perp}, \bar{A} = \bar{\alpha}] / \mathbf{Pr}[S_{\not\perp}] \\
&\ge \left(1 - \frac{t_0 + \Delta/2}{k}\right) \left(\frac{\sum_{\bar{\alpha} \in G \cup I} \mathbf{Pr}[S_{\not\perp}, \bar{A} = \bar{\alpha}]}{\mathbf{Pr}[S_{\not\perp}]}\right) \\
&= \left(1 - \frac{t_0 + \Delta/2}{k}\right) \cdot \frac{\mathbf{Pr}[S_{\not\perp}] - \mathbf{Pr}[S_{\not\perp}, \bar{A} \in B]}{\mathbf{Pr}[S_{\not\perp}]} \\
&= \left(1 - \frac{t_0 + \Delta/2}{k}\right) \cdot \left(1 - \frac{\mathbf{Pr}[S_{\not\perp}, \bar{A} \in B]}{\mathbf{Pr}[S_{\not\perp}]}\right).
\end{aligned}$$

Observe that $\mathbf{Pr}[S_{\not\perp}, \bar{A} \in B] \leq \mathbf{Pr}[S_{\not\perp} | \bar{A} \in B] * \mathbf{Pr}[\bar{A} \in B] \leq \rho^{\Delta/2} \cdot 1 = \rho^{\Delta/2}$, and $\mathbf{Pr}[S_{\not\perp}] \geq \mathbf{Pr}[S_{\not\perp}, \bar{A} \in G] \geq \epsilon$. Thus, we get that $1 - \mathbf{Pr}[S_{\not\perp}, \bar{A} \in B]/\mathbf{Pr}[S_{\not\perp}] \geq 1 - \rho^{\Delta/2}/\epsilon$, and the claim follows. □

Let A be the random variable denoting the first element of the pair produced by $\mathcal{E}_2$ conditioned on $\mathcal{E}_2$ producing a pair. We now write down the success probability of $\mathcal{E}_2$ in terms of the conditional probability that $\mathcal{E}_2$ produces a correct pair given that it produces a pair $(\alpha, .)$ for a fixed $\alpha \in \{0,1\}^n$.

$$\begin{aligned}\mathbf{Pr}[\mathcal{E}_2 \textit{ succeeds}] &= \mathbf{Pr}[S_c | S_{\not\perp}] \\ &= \sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[\mathcal{E}_2 \textit{ succeeds on } A | A = \alpha, S_{\not\perp}] \cdot \mathbf{Pr}[A = \alpha | S_{\not\perp}] \\ &= \sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[\mathcal{E}_2 \textit{ succeeds on } A | A = \alpha, S_{\not\perp}] \cdot \mathcal{D}_2(\alpha) \qquad (1)\end{aligned}$$

where $\mathcal{D}_2$ is a distribution defined as $\mathcal{D}_2(\alpha) = \mathbf{Pr}[A = \alpha | S_{\not\perp}]$.

Note the similarity between the distribution $\mathcal{D}_2$ and distribution $\mathcal{D}$ of the previous section. $\mathcal{D}$ was sampled by producing a randomly chosen element from a randomly chosen tuple in G. Here we allow tuples to be chosen from any G_i but we weigh down the contribution of the tuple by a factor of ρ^i. In other words, $\mathcal{D}_2$ can be sampled in the following manner: Pick a random tuple $\bar{\alpha} \in \{0,1\}^{nk}$, let $\bar{\alpha} \in G_i$, output a randomly chosen element of the tuple with probability ρ^i.

Comparing $\mathcal{D}_2$ and $\mathcal{U}$ We will show that $\mathcal{D}_2$ is statistically close to the uniform distribution $\mathcal{U}$.

Claim 3 $Dist(\mathcal{D}_2, \mathcal{U}) < 0.6\sqrt{\frac{\log 1/\epsilon}{k}}$.

Proof. To sample from $\mathcal{D}_2$, pick $(\alpha_1, \ldots, \alpha_k) \leftarrow \mathcal{G}$, pick a random $i \in [k]$ and output α_i, where $\mathcal{G}$ is a distribution on k-tuples such that $\mathcal{G}(\bar{\alpha})$ is the conditional probability that $\mathcal{E}_2$ outputs the randomly chosen element from $\bar{\alpha}$ given that $\mathcal{E}_2$ produces a pair. More specifically, given $\bar{\alpha} \in G_i$,

$$\mathcal{G}(\bar{\alpha}) = \frac{\rho^i}{|G_0| + \rho\ |G_1| + \ldots + \rho^{k(1-(1-\gamma)\delta)}\ |G_{k(1-(1-\gamma)\delta)}|} \leq \frac{1}{|G_0|} \leq \frac{1}{\epsilon \cdot 2^{nk}}.$$

By Corollary 1, we get the conclusion of the Claim. □

Comparing $\mathcal{E}_3$ and $\mathcal{E}_2$ To continue, we need the following definitions.

Definition 5. *Given $\alpha \in \{0,1\}^n$ and a k-tuple $(\alpha_1, \ldots, \alpha_k) \in (\{0,1\}^n)^k$, let $h(\alpha, (\alpha_1, \ldots, \alpha_k)) = \{i : \alpha_i = \alpha\}$. Given a k-tuple $(\alpha_1, \ldots, \alpha_k) \in (\{0,1\}^n)^k$ and solver $\bar{C}$, let $l(\alpha_1, \ldots, \alpha_k) = \{i : \neg R((x_i, \alpha_i), \bar{C}(x_1, \ldots, x_k)_i)\}$*

In other words, for a given element and tuple, h denotes the subset of indices where the element is present, and, for a given tuple, l denotes the subset of indices where $\bar{C}$ is incorrect. Consider the following two quantities:

$$X_\alpha = \underbrace{\sum_{\bar{\alpha}\in G} |h(\alpha,\bar{\alpha}) \cap l(\bar{\alpha})|}_{M_\alpha} + \underbrace{\sum_{\bar{\alpha}\in\{0,1\}^{nk}-G} |h(\alpha,\bar{\alpha}) \cap l(\bar{\alpha})| \cdot \rho^{|l(\bar{\alpha})|-t_0}}_{N_\alpha}$$

$$Y_\alpha = \sum_{\bar{\alpha}\in G} |h(\alpha,\bar{\alpha}) - h(\alpha,\bar{\alpha}) \cap l(\bar{\alpha})| + \sum_{\bar{\alpha}\in\{0,1\}^{nk}-G} |h(\alpha,\bar{\alpha}) - h(\alpha,\bar{\alpha}) \cap l(\bar{\alpha})| \cdot \rho^{|l(\bar{\alpha})|-t_0}$$

It is easy to see that

$$\mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{L}}] = \frac{Y_\alpha}{X_\alpha + Y_\alpha} = \frac{Y_\alpha}{M_\alpha + N_\alpha + Y_\alpha}. \tag{2}$$

Experiment $\mathcal{E}_3$ is mostly the same as $\mathcal{E}_2$, except when, for a randomly chosen tuple $\bar{\alpha} \in \{0,1\}^{nk} - G$ (line 1), the randomly chosen index i (line 2) lands in the subset $l(\bar{\alpha})$ of indices on which $\bar{C}$ is incorrect. Here $\mathcal{E}_2$ only outputs the pair with probability $\rho^{|l(\bar{\alpha})|-t_0}$ (instead of $\rho^{|l(\bar{\alpha})|-t_0-1}$ as in $\mathcal{E}_3$). Thus we have

$$\mathbf{Pr}[\mathcal{E}_3 \text{ succeeds on } A \mid A = \alpha, S_{\not{L}}] = \frac{Y_\alpha}{M_\alpha + N_\alpha/\rho + Y_\alpha}. \tag{3}$$

Finally, using (2) and (3), we get:

$$\begin{aligned}\frac{\mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{L}}]}{\mathbf{Pr}[\mathcal{E}_3 \text{ succeeds on } A \mid A = \alpha, S_{\not{L}}]} &= \frac{M_\alpha + N_\alpha/\rho + Y_\alpha}{M_\alpha + N_\alpha + Y_\alpha} \\ &\le \frac{M_\alpha + N_\alpha + Y_\alpha}{\rho \cdot (M_\alpha + N_\alpha + Y_\alpha)} \\ &= 1/\rho. \end{aligned} \tag{4}$$

Analyzing $\mathcal{C}$ We first note that the subset H of α's for which the above solver does not produce an answer (or produces $\perp$) is small. Consider the following two claims:

Claim 4 *Let $H \subseteq \{0,1\}^n$ be such that, for every $\alpha \in H$, TRS produces an answer with probability $< \epsilon/4$. Then $|H| < \frac{\gamma\delta}{4} \cdot 2^n$.*

Proof. For the sake of contradiction, assume that $|H| \ge \frac{\gamma\delta}{4} \cdot 2^n$. For a randomly chosen tuple $\bar{\alpha} = (\alpha_1, \ldots, \alpha_k)$, the expected number of α_i's from H is $\gamma\delta k/4$. By Chernoff bounds, all but $e^{-\frac{\gamma\delta k}{64}}$ fraction of tuples $\bar{\alpha}$ will contain at least $\gamma\delta k/8$ elements from H.

For a random $\alpha \in H$, consider the distribution on tuples $\bar{\alpha}$ induced by lines 3–5 of Solver 2. That is, $\bar{\alpha}$ is sampled by picking independently uniformly at random $\alpha \in H$, location $i \in [k]$, and $\alpha_1 \ldots \alpha_{k-1} \in \{0,1\}^n$, and producing $\bar{\alpha} = (\alpha_1, \ldots, \alpha_{i-1}, \alpha, \alpha_i, \ldots, \alpha_{k-1})$. Observe that every tuple $\bar{\alpha}'$ containing exactly s

elements from H will be assigned by this distribution probability exactly $\frac{4s}{\gamma\delta k}$ times the probability of $\bar{\alpha}'$ under the uniform distribution. So the probability of sampling tuples in G which have more than $\frac{\gamma\delta k}{8}$ elements from H is at least $\frac{\epsilon - e^{-\frac{\gamma\delta k}{64}}}{2} \geq \frac{\epsilon}{4}$, since $\epsilon = 2 \cdot e^{-\frac{\gamma^2\delta^2 k}{64}}$. This means that for a random $\alpha \in H$, a single iteration of the subroutine TRS of Solver 2 will produce a definite answer with probability at least $\epsilon/4$ (note that TRS always produces an answer when $\bar{\alpha}' \in G$). By averaging, there exists a particular $\alpha_0 \in H$ for which TRS succeeds in producing an answer with probability at least $\epsilon/4$. □

Claim 5 *For every* $\alpha \in \{0,1\}^n - H$, $\mathbf{Pr}[\mathcal{C}(x) \neq \perp] > 1 - e^{-n}$.

Proof. From the previous claim we know that for any $\alpha \in \{0,1\}^n - H$, the subroutine TRS produces an answer with probability at least $\epsilon/4$. So, the probability that Solver 2 fails to produce a definite answer on this input α within *timeout* iterations is at most $(1-\epsilon/4)^{\frac{4n}{\epsilon}} \leq e^{-n}$. □

The similarity between solver $\mathcal{C}$ and Experiment $\mathcal{E}_3$ yields the following useful fact:

$$\mathbf{Pr}[R((x,\alpha),\mathcal{C}(x)) | \mathcal{C}(x) \neq \perp] = \mathbf{Pr}[\mathcal{E}_3 \text{ succeeds on } A \mid A = \alpha, S_{\not\perp}]. \tag{5}$$

We now analyze the success probability of the solver $\mathcal{C}$. The probability is over uniformly random $\alpha \in \{0,1\}^n$ and its internal randomness.

$$\begin{aligned}
\mathbf{Pr}[\mathcal{C} \text{ succeeds}] &= \frac{1}{2^n} \sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[R((x,\alpha),\mathcal{C}(x)) \wedge \mathcal{C}(x) \neq \perp] \\
&= \sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[R((x,\alpha),\mathcal{C}(x)) \mid \mathcal{C}(x) \neq \perp] * \mathbf{Pr}[\mathcal{C}(x) \neq \perp] * \mathcal{U}(\alpha) \\
&= \sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[\mathcal{E}_3 \text{ succeeds on } A \mid A = \alpha, S_{\not\perp}] * \mathbf{Pr}[\mathcal{C}(x) \neq \perp] * \mathcal{U}(\alpha) \\
&\qquad \text{(from (5))} \qquad (6)
\end{aligned}$$

Let $H \subseteq \{0,1\}^n$ be the set from Claim 4. Let $\bar{H}$ be the complement of H in the set $\{0,1\}^n$. By Claim 5 and Eq. (4), we get that for every $\alpha \in \bar{H}$,

$$\begin{aligned}
&\mathbf{Pr}[\mathcal{E}_3 \text{ succeeds on } A \mid A = \alpha, S_{\not\perp}] * \mathbf{Pr}[\mathcal{C}(x) \neq \perp] * \mathcal{U}(\alpha) \geq \\
&\qquad (1 - e^{-n})\ \rho\ \mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not\perp}] * \mathcal{U}(\alpha).
\end{aligned} \tag{7}$$

Comparing $\mathcal{C}$ and $\mathcal{E}_2$ We can now compare the success probabilities of Experiment $\mathcal{E}_2$ and the solver $\mathcal{C}$.

Claim 6 $\mathbf{Pr}[\mathcal{C}$ *succeeds*$] \geq \mathbf{Pr}[\mathcal{E}_2$ *succeeds*$] - \left(Dist(\mathcal{U},\mathcal{D}_2) + (1-\rho) + \rho \cdot e^{-n} + \frac{\gamma\delta}{4}\right)$

Proof. Using the lower bound from (7), we can lower-bound $\mathbf{Pr}[\mathcal{C} \text{ succeeds}]$ as follows:

$$\mathbf{Pr}[\mathcal{C} \text{ succeeds}] \geq \rho(1-e^{-n}) \sum_{\alpha \in \bar{H}} \mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{\perp}}] * \mathcal{U}(\alpha).$$

Next, observe that

$$\sum_{\alpha \in \bar{H}} \mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{\perp}}] * \mathcal{U}(\alpha) \geq$$

$$\sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{\perp}}] * \mathcal{U}(\alpha) - \gamma\delta/4,$$

since $\sum_{\alpha \in H} \mathcal{U}(\alpha) < \gamma\delta/4$. Expressing $\mathcal{U}(\alpha)$ as $(\mathcal{U}(\alpha) - \mathcal{D}_2(\alpha)) + \mathcal{D}_2(\alpha)$, we can rewrite

$$\sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{\perp}}] * \mathcal{U}(\alpha)$$

as

$$\sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{\perp}}] * \mathcal{D}_2(\alpha) +$$

$$\sum_{\alpha \in \{0,1\}^n} \mathbf{Pr}[\mathcal{E}_2 \text{ succeeds on } A \mid A = \alpha, S_{\not{\perp}}] * (\mathcal{U}(\alpha) - \mathcal{D}_2(\alpha)).$$

The first summand is exactly $\mathbf{Pr}[\mathcal{E}_2 \text{ succeeds}]$. The second summand can be lower-bounded by restricting the summation to those $\alpha \in \{0,1\}^n$ where $\mathcal{U}(\alpha) < \mathcal{D}_2(\alpha)$, and observing that the resulting expression is at least $-Dist(\mathcal{U}, \mathcal{D}_2)$.

Putting it all together, we get that

$$\mathbf{Pr}[\mathcal{C} \text{ succeeds}] \geq \rho(1-e^{-n})(\mathbf{Pr}[\mathcal{E}_2 \text{ succeeds}] - Dist(\mathcal{U}, \mathcal{D}_2) - \gamma\delta/4).$$

Rearranging the terms on the right-hand side yields the claim. $\square$

The previous claim and Claim 2 yield the following final result.

Claim 7 $\mathbf{Pr}[\mathcal{C}$ *succeeds*$] \geq (1-\delta) + \frac{\gamma\delta}{32}$.

Proof. Indeed, we have

$$\mathbf{Pr}[\mathcal{C} \textit{ succeeds}] \geq \left(1 - \frac{t_0 + \Delta/2}{k}\right)\left(1 - \frac{\rho^{\Delta/2}}{\epsilon}\right) - \left(Dist(\mathcal{U}, \mathcal{D}_2) + (1-\rho) + \rho \cdot e^{-n} + \frac{\gamma\delta}{4}\right). \quad (8)$$

For $\rho = 1 - \frac{\gamma\delta}{16}, \epsilon = 2e^{-\frac{\gamma^2\delta^2 k}{64}}, \Delta = \gamma\delta k$ and $t_0 = (1-\gamma)\delta k$, we get $1 - (t_0 + \Delta/2)/k = 1 - \delta + \gamma\delta/2$ and $1 - \rho^{\Delta/2}/\epsilon \geq 1 - e^{-\gamma^2\delta^2 k/64}/2$. By Claim 3, we have that $Dist(\mathcal{U}, \mathcal{D}_2) \leq \gamma\delta/8$. So we can lowerbound the right-hand side of Eq. (8) by

$$1-\delta-(1-\delta)e^{-\gamma^2\delta^2 k/64}/2 + (1-e^{-\gamma^2\delta^2 k/64}/2)\gamma\delta/2 - (\gamma\delta/8 + \gamma\delta/16 + \epsilon^{-n} + \gamma\delta/4),$$

which is at least $1 - \delta + \gamma\delta/32$, for sufficiently large n. $\square$

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 ([18]). 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, analogously to how [6] improved the bounds from [4]? Finally, we would like to find more applications of our results, to such problems as making strong secret agreement protocols from weak ones ([9]).

References

1. Aaronson, S.: Limitations of quantum advice and one-way communication. In: Proceedings of the Nineteenth Annual IEEE Conference on Computational Complexity. (2004) 320–332
2. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using hard AI problems for security. In: EUROCRYPT. (2003) 294–311
3. Babai, L., Fortnow, L., Nisan, N., Wigderson, A.: BPP has subexponential time simulations unless EXPTIME has publishable proofs. Computational Complexity **3** (1993) 307–318
4. Bellare, M., Impagliazzo, R., Naor, M.: 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) 374–383
5. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Annals of Mathematical Statistics **23** (1952) 493–509
6. Canetti, R., Halevi, S., Steiner, M.: Hardness amplification of weakly verifiable puzzles. In: TCC. (2005) 17–33
7. Gal, A., Halevi, S., Lipton, R., Petrank, E.: Computing from partial solutions. In: Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity. (1999) 34–45
8. Goldreich, O., Nisan, N., Wigderson, A.: On Yao's XOR-Lemma. Electronic Colloquium on Computational Complexity (TR95-050) (1995)
9. Holenstein, T.: Key agreement from weak bit agreement. In: Proceedings of the 37th ACM Symposium on Theory of Computing. (2005) 664–673
10. Impagliazzo, R.: Hard-core distributions for somewhat hard problems. In: Proceedings of the Thirty-Sixth Annual IEEE Symposium on Foundations of Computer Science. (1995) 538–545
11. Impagliazzo, R., Jaiswal, R., Kabanets, V.: Approximately list-decoding direct product codes and uniform hardness amplification. In: Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science (FOCS06). (2006) 187–196
12. Impagliazzo, R., Wigderson, A.: 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) 220–229
13. Klivans, A.R.: On the derandomization of constant depth circuits. In: Proceedings of the 5th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM'01). (2001)

14. Klauck, H., Spalek, R., de Wolf, R.: Quantum and classical strong direct product theorems and optimal time-space tradeoffs. In: Proceedings of the Forty-Fifth Annual IEEE Symposium on Foundations of Computer Science. (2004) 12–21
15. Levin, L.A.: One-way functions and pseudorandom generators. Combinatorica **7**(4) (1987) 357–363
16. Nisan, N., Rudich, S., Saks, M.: Products and help bits in decision trees. In: Proceedings of the Thirty-Fifth Annual IEEE Symposium on Foundations of Computer Science. (1994) 318–329
17. Parnafes, I., Raz, R., Wigderson, A.: Direct product results and the GCD problem, in old and new communication models. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing. (1997) 363–372
18. Pass, R., Venkitasubramaniam, M.: An efficient parallel repetition theorem for arthur-merlin games. In: STOC'07 (To appear). (2007)
19. Pietrzak, K., Wikstrom, D.: Parallel repetition of computationally sound protocols revisited. In: TCC'07 (To appear). (2007)
20. Raz, R.: A parallel repetition theorem. SIAM Journal on Computing **27**(3) (1998) 763–803
21. Shaltiel, R.: Towards proving strong direct product theorems. In: Proceedings of the Sixteenth Annual IEEE Conference on Computational Complexity. (2001) 107–119
22. Trevisan, L.: List-decoding using the XOR lemma. In: Proceedings of the Forty-Fourth Annual IEEE Symposium on Foundations of Computer Science. (2003) 126–135
23. Yao, A.C.: Theory and applications of trapdoor functions. In: Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science. (1982) 80–91