General Hardness Amplification of Predicates and Puzzles

(extended abstract)[*]

Thomas Holenstein[1][**] and Grant Schoenebeck[2][***]

[1] Department of Computer Science, ETH Zurich, Switzerland,
thomas.holenstein@inf.ethz.ch,
WWW home page: http://www.complexity.ethz.ch/people/holthoma

[2] Department of Computer Science, Princeton University, Princeton NJ 08544, USA,
gschoene@princeton.edu,
WWW home page: http://www.cs.princeton.edu/~gschoene

Abstract. We give new proofs for the hardness amplification of efficiently samplable predicates and of weakly verifiable puzzles which generalize to new settings. More concretely, in the first part of the paper, we give a new proof of Yao's XOR-Lemma that additionally applies to related theorems in the cryptographic setting. Our proof seems simpler than previous ones, yet immediately generalizes to statements similar in spirit such as the extraction lemma used to obtain pseudo-random generators from one-way functions [Håstad, Impagliazzo, Levin, Luby, SIAM J. on Comp. 1999].

In the second part of the paper, we give a new proof of hardness amplification for weakly verifiable puzzles, which is more general than previous ones in that it gives the right bound even for an arbitrary monotone function applied to the checking circuit of the underlying puzzle.

Both our proofs are applicable in many settings of interactive cryptographic protocols because they satisfy a property that we call "non-rewinding". In particular, we show that any weak cryptographic protocol whose security is given by the unpredictability of single bits can be strengthened with a natural information theoretic protocol. As an example, we show how these theorems solve the main open question from [Halevi and Rabin, TCC2008] concerning bit commitment.

1 Introduction

In this paper, we study two scenarios of hardness amplification. In the first scenario, one is given a predicate $P(x)$, which is somewhat hard to compute given x. More concretely: $\Pr[A(x) = P(x)] \leq 1 - \frac{\delta}{2}$ for any A in some given

[*] A full version of this paper is available [22].

[**] Work was done while the author was at Microsoft Research, Silicon Valley Campus

[***] Work was partially done while author was a summer intern at Microsoft Research Silicon Valley Campus and partially supported by a National Science Foundation Graduate Fellowship.

complexity class, where typically δ is not too close to 1 but at least polynomially big (say, $\frac{1}{\text{poly}(n)} < \delta < 1 - \frac{1}{\text{poly}(n)}$). One then aims to find a predicate which is even harder to compute.

In the second scenario, one is given a computational search problem, specified by some relation $R(x, y)$. One then assumes that no algorithm of a certain complexity satisfies $\Pr[(x, A(x)) \in R] > 1 - \delta$, and again, is interested in finding relations which are even harder to satisfy. It is sometimes the case that R may only be efficiently computable given some side information generated while sampling x. Such problems are called "weakly verifiable puzzles".

Our aim is to give proofs for theorems in both scenarios which are both simple and versatile. In particular, we will see that our proofs are applicable in the interactive setting, where they give stronger results than those previously known.

1.1 Predicates

Overview and previous work. Roughly speaking, Yao's XOR-Lemma [39] states that if a predicate $P(x)$ is somewhat hard to compute, then the k-wise XOR $P^{\oplus k}(x_1, \dots, x_k) := P(x_1) \oplus \cdots \oplus P(x_k)$ will be even harder to compute. While intuitive, such statements are often somewhat difficult to prove. The first proof of the above appears to be by Levin [31] (see also [11]). In some cases, even stronger statements are needed: for example, the extraction lemma states that one can even extract several bits out of the concatenation $P(x_1)P(x_2)\dots P(x_k)$, which look pseudorandom to a distinguisher given $x_1, \dots, x_k$. Proving this statement for tight parameters is considered the technically most difficult step in the original proof that one-way functions imply pseudorandom generators [17]. Excluding this work, the easiest proof available seems to be based on Impagliazzo's hard-core set theorem [23], more concretely the uniform version of it [19, 1]. A proof along those lines is given in [20, 13]. Similar considerations are true for the more efficient proof that one-way functions imply pseudorandom generators given by Haitner et al.[15].

Contributions of this paper. In this paper, we are concerned with statements of a similar nature as (but which generalize beyond) Yao's XOR-Lemma. We give a new theorem, which is much easier to prove than the hard-core set theorem, and which is still sufficient for all the aforementioned applications.

Our main observation can be described in relatively simple terms. In the known proof based on hard-core sets ([23, 19]), the essential statement is that there is a large set S, such that for $x \in S$ it is computationally difficult to predict $P(x)$ with a non-negligible advantage over a random guess. Proving the existence of the set S requires some work (basically, boosting, as shown in [30]). We use the idea that the set S can be made *dependent* on the circuit which attempts to predict P. The existence of a hard set S for a particular circuit is a much easier fact to show (and occurs as a building block in some proofs of the hard-core theorem). For our idea to go through, S has to be made dependent on some of the inputs to C as well as some other fixed choices. This technique of switching quantifiers resembles a statement in [2], where Impagliazzo's hard-core set theorem is used to show that in some definitions of pseudo-entropy it is also possible to switch quantifiers.

Besides being technically simpler, making the set S dependent on C has an additional advantage. For example, consider a proof of the XOR Lemma. To get a contradiction, a circuit C is assumed which does well in predicting the XOR, and a circuit D for a single instance is built from C. On input x, D calls C as a subroutine several times, each time "hiding" x as one of the elements of the input. Using our ideas, we can ensure that x is hidden always in the same place i, and even more, the values of the inputs $x_1, \ldots, x_{i-1}$ are constant and independent of x. This property, which we call non-rewinding, is useful in the case one wants to amplify the hardness of interactive protocols.

We remark that in this paper we are not concerned with efficiency of XOR-Lemmas in the sense of derandomizing them (as in, e.g., [28, 24, 26]).

1.2 Weakly Verifiable Puzzles

Overview and Previous Work. The notion of weakly verifiable puzzles was introduced by Canetti et al. [4]. A weakly verifiable puzzle consists of a sampling method, which produces an instance x together with a circuit $\Gamma(y)$, checking solutions. The task is, given x but not necessarily Γ, to find a string y for which $\Gamma(y) = 1$. One-way functions are an example: $\Gamma(y)$ just outputs 1 if $f(y) = x$ (since Γ depends on the instance it can contain x). However, weakly verifiable puzzles are more general, since Γ is not given at the time y has to be found.

Canetti et al. show that if no efficient algorithm finds solutions with probability higher than δ, then any efficient algorithm finds k solutions simultaneously with probability at most $\delta^k + \epsilon$, for some negligible ϵ. This result was strengthened by [25], showing that requiring some $\delta' > \delta + 1/\operatorname{poly}(n)$ fraction of correct answers already makes efficient algorithms fail, if k is large enough. Independently of the current work, Jutla [29] improved their bound to make it match the standard Chernoff bound. A different strengthening was given in [16], where it was noted that the algorithm in [4] has an additional property which implies that it can be applied in an interactive cryptographic setting, also they studied how much easier solving a weakly verifiable puzzle becomes if one simply asks for a single correct solution from k given puzzles. Also independently of our work, Chung et al. [6] give a proof for the threshold case (similar to Jutla) which is also applicable in an interactive setting; however, their parameters are somewhat weaker than the ones given by most other papers. Finally, [9] gives yet another strengthening: they allow a weakly verifiable puzzle to have multiple solutions indexed by some element q, and the adversary is allowed to interactively obtain some of them. They then study under what conditions the hardness is amplified in this setting.

Contributions of this paper. In this work, we present a theorem which unifies and strengthens the results given in [4, 16, 25, 29, 6]: assume a monotone function $g : \{0,1\}^k \to \{0,1\}$ specifies which subpuzzles need to be solved in order to solve the resulting puzzle (i.e., if $c_1, \dots, c_k$ are bits where c_i indicates that a valid solution for puzzle i was found, then $g(c_1, \dots, c_k) = 1$ iff this is sufficient to give a valid solution for the overall case.) Our theorem gives a tight bound for any such g (in this sense, previous papers considered only threshold functions for g). Furthermore, as we will see our proof is also applicable in an interactive setting (the proof given in [25, 29] does not have this property). Our proof is heavily inspired by the one given in [4].

1.3 Strengthening Cryptographic Protocols

Overview and Previous Work. Consider a cryptographic protocol, such as bit commitment. Suppose that a non-perfect implementation of such a protocol is given, which we would like to improve. For example, assume that a cheating receiver can guess the bit committed to with some probability, say 3/5. Furthermore, suppose that a cheating sender can open the commitment in two ways with some probability, say 1/5. Can we use this protocol to get a stronger bit commitment protocol?

Such questions have been studied in various forms both in the information theoretic and the computational model [8, 7, 10, 19, 21, 38, 16].

However, all of the previous computational work except [16] focused on the case where the parties participating in the protocol are at least semi-honest, i.e., they follow the protocol correctly (this is a natural assumption in the case for the work on key agreement [10, 19, 21], as in this case the participating parties can be assumed to be honest). An exception to this trend was the work by Halevi and Rabin [16], where it was shown that for *some* protocols, the information theoretic bounds also apply computationally.

The above are results in case where the protocol is repeated *sequentially.* The case where the protocol is repeated in parallel is more complicated [3, 35, 34, 18, 12, 5].

Contributions of this paper. We explicitly define "non-rewinding" (which was, however, pointed to in [16]) which helps to provide a sufficient condition for transforming complexity theoretic results into results for cryptographic protocols. Using, the above results, and specifically that the above results are non-rewindable, we show that we can strengthen any protocol in which the security goal is to make a bit one party has unpredictable to the other party, in the case where an information theoretic analogue can be strengthened. We also study interactive weakly verifiable puzzles (as has been done implicitly in [16]), and show that natural ways to amplify the hardness of these work.

We only remark that our proof is applicable to parallel repetition for non-interactive (two-round) protocols (e.g. CAPTCHAs).

Due to space restrictions, many of the proofs and even some of the formal statements of theorems have been omitted. We encourage the interested reader to look at the full version of this paper [22].

2 Preliminaries

Definition 1. *Consider a circuit C which has a tuple of designated input wires labeled $y_1, \ldots, y_k$. An oracle circuit $D(\cdot)$ with calls to C is* non-rewinding *if there is a fixed i and fixed strings y_1^* to y_{i-1}^* such that for any input y to D, all calls to C use inputs $(y_1^*, \ldots, y_{i-1}^*, y)$ on the wires labeled $y_1, \ldots, y_i$.*

Definition 2. *Let C be a circuit which has a block of input wires labeled x. An oracle circuit D which calls C (possibly several times) treats x obliviously if the input x to D is forwarded to C directly, and not used in any other way in D.*

We say that an event happens *almost surely* if it has probability $1-2^{-n}\operatorname{poly}(n)$.

We denote by $[m]$ the set $\{1, \ldots, m\}$. The density of a set $S \subseteq \{0,1\}^n$ is $\mu(S) = \frac{|S|}{2^n}$. We sometimes identify a set S with its characteristic function $S : \{0,1\}^n \to \{0,1\}$. We often denote a tuple $(x_1, x_2, \ldots, x_k)$ by $x^{(k)}$.

If a distribution μ over some set is given, we write $x \leftarrow \mu$ to denote that x is chosen according to μ. We sometimes identify sets with the uniform distribution over them. We let μ_δ be the Bernoulli distribution over $\{0,1\}$ with parameter δ, i.e., $\Pr_{x \leftarrow \mu_\delta}[x = 1] = \delta$. Furthermore, μ_δ^k is the distribution over $\{0,1\}^k$ where each bit is i.i.d. according to μ_δ.

When two interactive algorithms A and B are given, we will denote by $\langle A, B\rangle_A$ the output A has in an interaction with B, and by $\langle A, B\rangle_B$ the output which B has. We sometimes consider probabilities like $\Pr[\langle A, B\rangle_A = \langle A, B\rangle_B]$, in which case the probability is over random coins of A and B (if any), but they are chosen the same on the left and the right hand side.

3 Efficiently Samplable Predicates

3.1 Single Instance

Informal Discussion. Fix a predicate $P : \{0,1\}^n \to \{0,1\}$ and a circuit $C(x, b, r)$ which takes an arbitrary $x \in \{0,1\}^n$, a bit $b \in \{0,1\}$, and some randomness r as input. We may think of C as a circuit which tries to distinguish the case $b = P(x)$ from the case $b = 1 - P(x)$. Our idea is to identify a set S for which we can show the following:

1. If x is picked randomly from S, then $\Pr[C(x, P(x), r) = 1] \approx \Pr[C(x, 1 - P(x), r) = 1]$.
2. C can be used to predict $P(x)$ for a uniform random x correctly with probability close to $1 - \frac{1}{2}\mu(S)$

On an informal level, one could say that S explains the hardness of computing P from C's point of view: for elements from S the circuit just behaves as a uniform random guess, on the others it computes (or, more accurately, *helps* to compute) P. Readers familiar with Impagliazzo's hardcore lemma will notice the similarity: Impagliazzo finds a set which explains the computational difficulty of a predicate for *any* circuit of a certain size. Thus, in this sense Impagliazzo's theorem is stronger. The advantage of ours is that the proof is technically simpler, and that it can be used in the interactive setting (see Section 3.3) which seemingly comes from the fact that it helps to build non-rewinding proofs.

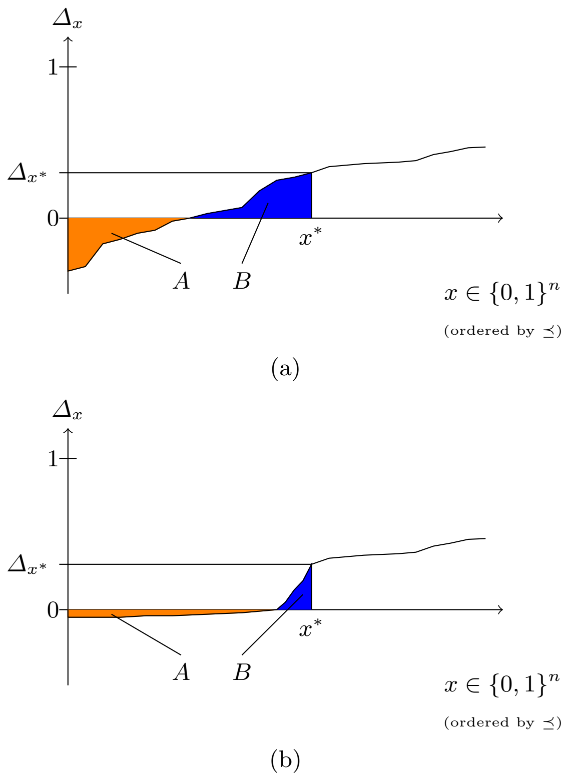

Fig. 1. Intuition for the proof of Theorem 1. In both pictures, on the vertical axis, the advantage of the circuit in guessing right over a random guess is depicted. The elements are then sorted according to this quantity. The point x^* is chosen such that the area of A is slightly smaller than the area of B.

The Theorem. The following theorem formalizes the above discussion. It will find S by producing a circuit which recognizes it, and also produces a circuit Q which uses C in order to predict P.

Theorem 1. *Let $P : \{0,1\}^n \to \{0,1\}$ be a computable predicate. There is an algorithm* Gen *which takes as input a randomized circuit $C(x, b, r)$ and a parameter ϵ, and outputs two deterministic circuits Q and S, both of size $\mathrm{size}(C) \cdot \mathrm{poly}(n, \frac{1}{\epsilon})$, as well as $\delta \in [0,1]$, such that almost surely the following holds:*

Large Set: *$S(x, P(x))$ recognizes a set $S^* = \{x|S(x, P(x)) = 1\}$ of density at least $\mu(S^*) \geq \delta$.*

Indistinguishability: *For the above set S^* we have*

$$\Big| \Pr_{x \leftarrow \{0,1\}^n, r}[C(x, P(x), r) = 1] - \Pr_{x \leftarrow \{0,1\}^n, r}[C(x, P'(x), r) = 1]\Big| \leq \epsilon, \qquad (1)$$

where $P'(x) := P(x) \oplus S(x)$, i.e., P' is the predicate which equals P outside S and differs from P within S.

Predictability: *Q predicts P well: $\Pr_{x \leftarrow \{0,1\}^n}[Q(x) = P(x)] \geq 1 - \frac{\delta}{2}$.*

Additionally, these algorithms have the following properties:

1. *Unless $\delta = 1$ algorithm Q predicts slightly better:*[3] $\Pr[Q(x) = P(x)] \geq 1 - \frac{\delta}{2} + \frac{\epsilon}{4}$.
2. *If P is efficiently samplable (i.e., pairs $(x, P(x))$ can be generated in polynomial time),* Gen *runs in time* $\mathrm{poly}(n, \frac{1}{\epsilon})$.
3. Gen*, S, and Q can be implemented with oracle access to C only (i.e., they do not use the description of C).*
4. *When thought as oracle circuits, S and Q use the oracle C at most $\mathcal{O}(\frac{n}{\epsilon^2})$ times. Also, they both treat x obliviously, and their output only depends on the number of* 1*'s obtained from the oracle calls to C and, in case of S, the input $P(x)$.*

The proof uses no new techniques. For example, it is very similar to Lemma 2.4 in [19], which in turn is implicit in [31, 11] (see also Lemma 6.6 and Claim 7 on page 121 in [20]). Our main contribution here is to give the statement and to note that it is very powerful. The proof itself is only given in the full version of the paper [22], which we encourage the reader to view. It is only remarkable for how straight-forward it is (given the statement).

Proof Overview. We assume that overall $C(x, P(x), r)$ is more often 1 than $C(x, 1 - P(x), r)$. Make S the largest set for which the Indistinguishability property is satisfied as follows: order the elements of $\{0,1\}^n$ according to $\Delta_x := \Pr_r[C(x, P(x), r) = 1] - \Pr_r[C(x, 1 - P(x), r) = 1]$, and insert them into S sequentially until both $\Pr_{x \leftarrow S, r}[C(x, P(x), r) = 1] > \Pr_{x \leftarrow S, r}[C(x, 1 - P(x), r) = 1]$ and indistinguishability is violated. Then, it only remains to describe Q. For any $x \notin S$ note that $\Pr[C(x, P(x), r) = 1] - \Pr[C(x, 1 - P(x), r) = 1] \geq \epsilon$, as otherwise x could be added to S. Thus, for those elements $P(x)$ is the bit b for which $\Pr[C(x, b, r) = 1]$ is bigger. In this overview we assume that $\Pr[C(x, b, r) = 1]$ can be found exactly, so we let $Q(x)$ compute the probabilities for $b = 0$ and $b = 1$, and answer accordingly; we will call this rule the "Majority Rule". Clearly, $Q(x)$ is correct if $x \notin S$, and in order to get "predictability", we only need to argue that Q is not worse than a random guess on S.

Consider now Figure 1 (a), where the elements are ordered according to Δ_x. The areas depicted A and B are roughly equal, which follows by the way we

[3] This implies that $\delta \geq \frac{\epsilon}{2}$, which can always be guaranteed.

chose S (note that $\Pr_{x \leftarrow S,r}[C(x,P(x),r) = 1] - \Pr_{x \leftarrow S,r}[C(x,1-P(x),r) = 1] = \mathbf{E}_{x \leftarrow S}[\Delta_x]$).

At this point our problem is that the majority rule will give the incorrect answer for all elements for which $\Delta_x < 0$, and as shown in Figure 1 (b), this can be almost all of S, so that in general the above Q *does* perform worse than a random guess on S. The solution is to note that it is sufficient to follow the majority rule in case the gap is bigger than Δ_{x^*}. In the full proof we will see that if the gap is small so that $-\Delta_{x^*} \leq \Pr[C(x,0,r) = 1] - \Pr[C(x,1,r) = 1] \leq \Delta_{x^*}$ then a randomized decision works: the probability of answering $b = 0$ is 1 if the gap is $-\Delta_{x^*}$, the probability of answering $b = 0$ is 0 if the gap is Δ_{x^*}. When the gap is in between then the probability of answering $b = 0$ is linearly interpolated based on the value of the gap. So for example, if the gap is 0, then $b = 0$ with probability $\frac{1}{2}$.[4] A bit of thought reveals that this is exactly because the areas A and B in Figure 1 are almost equal.

In the full proof, we also show how to sample all quantities accurately enough (which is easy) and how to ensure that S is a set of the right size (which seems to require a small trick because Δ_x as defined above is not computable exactly, and so we actually use a different quantity for Δ_x). We think that the second is not really required for the applications later, but it simplifies the statement of the above theorem and makes it somewhat more intuitive.

3.2 Multiple instances

Informal Discussion. We explain our idea on an example: suppose we want to prove Yao's XOR-Lemma. Thus, we are given a predicate $P : \{0,1\}^n \rightarrow \{0,1\}$ which is somewhat hard to compute, i.e., $\Pr[C^{(1)}(x) = P(x)] < 1 - \frac{\delta}{2}$ for any circuit $C^{(1)}$ coming from some family of circuits (the superscript (1) should indicate that this is a circuit operating on a single instance). We want to show that any circuit $C^{(\oplus k)}$ from a related family predicts $P(x_1) \oplus \cdots \oplus P(x_k)$ from $(x_1, \ldots, x_k)$ correctly with probability very close to $\frac{1}{2}$, and aiming for a contradiction we now assume that a circuit $C^{(\oplus k)}$ exists which does significantly better than this is given.

As a first step, we transform $C^{(\oplus k)}$ into a circuit $C^{(k)}(x_1, b_1, x_2, b_2, \ldots, x_k, b_k)$ as follows: $C^{(k)}$ invokes $C^{(\oplus k)}(x_1, \ldots, x_k)$ and outputs 1 if the result equals $b_1 \oplus \cdots \oplus b_k$, otherwise it outputs 0. We see that we would like to show
$\Pr[C^{(k)}(x_1, P(x_1), \ldots, x_k, P(x_k)) = 1] \approx \frac{1}{2}$.

Here is the key idea: we apply Theorem 1 sequentially on every position i of $C^{(k)}$. Done properly, in each position one of the following happens: (a) we can use $C^{(k)}$ to predict $P(x)$ from x with probability at least $1 - \frac{\delta}{2}$, or (b) we find a large set S_i^* such that if $x_i \in S_i^*$, $C^{(k)}$ behaves roughly the same in case b_i equals $P(x_i)$ and in case b_i is a uniform random bit. If (a) happens at any point we get a contradiction and are done, so consider the case that (b) happens

[4] It may be instructive to point out another rule which does not work: if one produces a uniform random bit in case the gap is smaller than Δ_{x^*} then elements in the region marked A with negative gap larger than Δ_{x^*} are problematic.

k times. Recall now how $C^{(k)}$ was built from $C^{(\oplus k)}$: it compares the output of $C^{(\oplus k)}$ to $b_1 \oplus \cdots \oplus b_k$. If x_i lands in the large set for any i we can assume that b_i is a random bit (and it is very unlikely that this happens for no i). Then, $C^{(k)}$ outputs 1 exactly if $C^{(\oplus k)}$ correctly predicts a uniform random bit which is independent of the input to $C^{(\oplus k)}$. The probability such a prediction is correct is exactly $\frac{1}{2}$, and overall we get that $C^{(\oplus k)}$ is correct with probability close to $\frac{1}{2}$.

The theorem gives the formal statement for $C^{(k)}$, in the full version the transformation to $C^{(\oplus k)}$ is done as an example.

The Theorem. Fix a predicate $P : \{0,1\}^n \to \{0,1\}$ and a boolean circuit $C^{(k)}(x_1, b_1, \ldots, x_k, b_k)$. We are interested in the probability that the circuit outputs 1 in the following Experiment 1:

Experiment 1:

```
    $\forall i \in \{1,\ldots,k\} : x_i \leftarrow \{0,1\}^n$
    $\forall i \in \{1,\ldots,k\} : b_i := P(x_i)$
    $r \leftarrow \{0,1\}^*$
    **output** $C^{(k)}(x_1, b_1, \ldots, x_k, b_k, r)$
```

We will claim that there are large sets $S_1^*, \ldots, S_k^*$ with the property that for any x_i which falls into S_i^*, we can set b_i to a random bit and the probability of the experiment producing a 1 will not change much. However, we will allow the sets S_i^* to depend on the x_j and b_j for $j < i$; we therefore assume that an algorithm GenS is given which produces such a set on input $t_i = (x_1, b_1, \ldots, x_{i-1}, b_{i-1})$.

Experiment 2:

```
    **for** $i := 1$ to $k$ **do**
        $t_i := (x_1, b_1, \ldots, x_{i-1}, b_{i-1})$
        $S_i^* := \mathrm{GenS}(t_i)$
        $x_i \leftarrow \{0,1\}^n$
        **if** $x_i \in S_i^*$ **then** $b_i \leftarrow \{0,1\}$ **else** $b_i := P(x_i)$ **fi**
    **end for**
    $r \leftarrow \{0,1\}^*$
    **output** $C^{(k)}(x_1, b_1, \ldots, x_k, b_k, r)$
```

Theorem 2 essentially states the following: assume no small circuit can predict $P(x)$ from x with probability $1 - \frac{\delta}{2}$. For any fixed circuit $C^{(k)}$, any ϵ, and any k there is an algorithm GenS which produces sets S_i^* with $\mu(S_i^*) \geq \delta$ and such that the probability that Experiment 1 outputs 1 differs by at most ϵ from the probability that Experiment 2 outputs 1.

Theorem 2. *Let P be a computable predicate, $k, \frac{1}{\epsilon} \in \mathrm{poly}(n)$ parameters. There are two algorithms* Gen *and* GenS *as follows:* Gen *takes as input a randomized circuit $C^{(k)}$ and a parameter ϵ and outputs a deterministic circuit Q of size* $\mathrm{size}(C^{(k)}) \cdot \mathrm{poly}(n)$ *as well as $\delta \in [0,1]$.* GenS *takes as input a circuit $C^{(k)}$, a tuple t_i, and a parameter ϵ and outputs a deterministic circuit $S_{t_i}(x,b)$ of* $\mathrm{size}(C^{(k)}) \cdot \mathrm{poly}(n)$*. After a run of* Gen*, almost surely the following properties are satisfied:*

Large Sets: *For any value of $t_i := (x_1, b_1, \ldots, x_{i-1}, b_{i-1})$ the circuit $S_{t_i}(x_i, P(x_i))$ recognizes a set $S_i^* := \{x_i | S(t_i, x_i, P(x_i)) = 1\}$. The probability that in an execution of Experiment 2 we have $\mu(S_i^*) < \delta$ for any of the S_i^* which occur is at most ϵ.*

Indistinguishability: *Using sets $S_{t_i}^*$ as above in Experiment 2 gives*

$$\Big|\Pr[\textit{Experiment 1 outputs 1}] - \Pr[\textit{Experiment 2 outputs 1}\Big| \leq \epsilon. \qquad (2)$$

Predictability: *Q predicts P well:* $\Pr_{x \leftarrow \{0,1\}^n}[Q(x) = P(x)] \geq 1 - \dfrac{\delta}{2}$.

Additionally, these algorithms have the following properties:

1. *Unless $\delta = 1$ algorithm Q predicts slightly better:* $\Pr[Q(x) = P(x)] \geq 1 - \frac{\delta}{2} + \frac{\epsilon}{16k}$.
2. *If P is efficiently samplable (i.e., pairs $(x, P(x))$ can be generated in polynomial time),* Gen *and* GenS *run in time* poly(n).
3. Gen, GenS, *S_{t_i}, and Q can be implemented with oracle access to C only (i.e., they don't use the description of C).*
4. *When thought of as oracle circuits, S_{t_i} and Q use the oracle C at most $\mathcal{O}(\frac{k^2 n}{\epsilon^2})$ times. Also, they both treat x obliviously and are non-rewinding. Finally, their output only depends on the number of 1's obtained from the oracle calls to C and, in case of S_{t_i}, the input $P(x)$.*

The proof is given in the full version, but follows the informal discussion above. We encourage the interested reader to see the full version [22].

3.3 Cryptographic Protocols which output single bits

Again we start with an example: consider a slightly weak bit commitment protocol, where the receiver can guess the bit the sender committed to with probability $1 - \frac{\delta}{2}$. In such a case, we might want to strengthen the scheme. For example, in order to commit to a single bit b, we could ask the sender to first commit to two random bits r_1 and r_2, and then send $b \oplus r_1 \oplus r_2$ to the receiver. The hope is that the receiver has to guess both r_1 and r_2 correctly in order to find b, and so the protocol should be more secure.

In the case where the protocol has some defect that sometimes allows a sender to cheat, we might also want to consider the protocol where the sender commits twice to b, or, alternatively, that he commits to r_1, then to r_2, and sends both $b \oplus r_1$ and $b \oplus r_2$ to the receiver. In this case, one can hope that a cheating receiver still needs to break the protocol at least once, and that the security should not degrade too much.

Just how will the security change? We want to consider a scenario in which the security is information theoretic. We can do this by assuming that instead of the weak protocol, a trusted party distributes a bit X to the sender and some side information Z to the receiver. The guarantee is that for any f, $\Pr[f(Z) = X] \leq 1 - \frac{\delta}{2}$. In such a case, one can easily obtain bounds on the security of the above protocols, and the hope is that the same bounds hold in the computational case. The theorem below states that this is indeed true (for protocols where the security consists of hiding single bits).

We remark that while the two aforementioned examples of protocol composition are already handled in [16] (their result applies to any direct product and any XOR as above), Theorem 3 handles any information theoretic amplification protocol as long as it can be implemented efficiently.

Definition 3. *A pair (X, Z) of random variables over $\{0,1\} \times \mathcal{Z}$, where $\mathcal{Z}$ is any finite set, is δ-hiding if*

$$\max_{f:\mathcal{Z}\to\{0,1\}} \Pr[f(Z) = X] \leq 1 - \frac{\delta}{2}. \tag{3}$$

Theorem 3. *Let a cryptographic protocol (which we think of as "weak") $W = (A_W, B_W)$ be given in which A_W has as input a single bit c. Assume that there is a function δ such that for any polynomial time adversary B^*_W there is a negligible function ν such that*

$$\Pr_{x\leftarrow\{0,1\}}[\langle A_W(x), B^*_W\rangle_B = x] \leq 1 - \frac{\delta}{2} + \nu(n), \tag{4}$$

where the probability is also over the coins of A_W and B^*_W (if any).

Let further an information theoretic protocol $I = (A_I, B_I)$ be given. In I, A_I takes k input bits $(X_1, \ldots, X_k)$ and has a single output bit. Furthermore, assume that I is hiding in the sense that for k independent δ-hiding random variables (X_i, Z_i), any (information theoretic) adversary B^*_I, and for some function $\eta(k)$:

$$\Pr\begin{bmatrix}\langle A_I(X_1, \ldots, X_k), B^*_I(Z_1, \ldots, Z_k)\rangle_A = \\ \langle A_I(X_1, \ldots, X_k), B^*_I(Z_1, \ldots, Z_k)\rangle_B\end{bmatrix} < \frac{1}{2} + \eta(k). \tag{5}$$

Let $S = (A_S, B_S)$ be the protocol where A and B first execute $k(n)$ copies of W sequentially, where A uses uniform random bits as input. Then, they run a single execution of protocol I. In the execution to I, A uses his k input bits to the weak protocols as input. The output of A in S is the output of A in the execution of I. We also need that (A_I, B_I) and $k(n)$ are such that I can be run in time $\mathrm{poly}(n)$ *for $k = k(n)$.*

Then, for any polynomial time B^*_S there is a negligible function ν' such that

$$\Pr[\langle A_S, B^*_S\rangle_A = \langle A_S, B^*_S\rangle_B] \leq \frac{1}{2} + \eta(k) + \nu'(n) \ . \tag{6}$$

Proof. Let $x \in \{0,1\}^n$ be the concatenation of the randomness which A uses in an execution of the protocol W and his input bit c. We let $P : \{0,1\}^n \to \{0,1\}$ be the predicate which outputs $c = P(x)$.

In order to obtain a contradiction, we fix an adversary B^*_S for the protocol S which violates (6). We would like to apply Theorem 2. For this, we define $C^{(k)}(x_1, b_1, \ldots, x_k, b_k)$ as follows: $C^{(k)}$ first simulates an interaction of B^*_S with

A_S, where A_S uses randomness x_i in the ith invocation of the weak protocol W. After this, B_S^* is in some state in which it expects an invocation of the information theoretic protocol. $C^{(k)}$ simulates this information theoretic protocol, but it runs A_I with inputs $b_1, \dots, b_k$ instead of the actual inputs to the weak protocols. In the end, B_S^* produces a guess for the output bit of A_S, and $C^{(k)}$ outputs 1 if this guess equals the output of $A_I(b_1, \dots, b_k)$ in the simulation.

In Experiment 1 of Theorem 2, $b_i = P(x_i)$ is used, and so $C^{(k)}$ exactly simulates an execution of the protocol S. Since we assume that B_S^* contradicts (6), we see that the probability that $C^{(k)}$ outputs 1 in Experiment 1 is, for infinitely many n and some constant c at least $\frac{1}{2} + \eta(k) + n^{-c}$.

We now apply Theorem 2 on the circuit $C^{(k)}$ with parameter $n^{-c}/3$. This yields a parameter δ_{T2} (the subscript indicates that it is from Theorem 2). We claim that

$$\delta_{T2} \leq \delta \qquad\qquad \text{almost surely.} \tag{7}$$

To see this, we assume otherwise and obtain a contradiction. In Experiment 2, Let Γ_i be the communication produced by the weak protocol W in round i. Assuming all sets S_i^* in the execution are of size at least δ (this happens with probability at least $1 - n^{-c}/3$), the tuples (b_i, Γ_i) are δ-hiding random variables. Consequently, when the circuit $C^{(k)}$ simulates the information theoretic protocol I using bits b_i, it actually simulates it in an instance in which it was designed to be used. Since (5) holds for an arbitrary adversary in this case we get that

$$\begin{aligned}&\Pr[C^{(k)} \text{ outputs 1 in Experiment 2} | \text{No set } S_i^* \text{ was of measure less than } \delta] \\ &\quad \leq \frac{1}{2} + \eta(k).\end{aligned} \tag{8}$$

Therefore, the probability that $C^{(k)}$ outputs 1 in Experiment 2 is at most $\frac{1}{2} + \eta(k) + \frac{n^{-c}}{3}$, and using "indistinguishability" the probability that $C^{(k)}$ outputs 1 in Experiment 1 is at most $\frac{1}{2} + \eta(k) + \frac{2n^{-c}}{3}$. However, our assumption was that the probability that $C^{(k)}$ outputs 1 is at least $\frac{1}{2} + \eta(k) + n^{-c}$, and so almost surely Gen does not output such a big δ_{T2}, establishing (7).

Theorem 2 also provides us with a non-rewinding circuit Q which treats x obliviously and which satisfies "predictability". We explain how to use Q to break (4), the security property of the weak protocol W.

Since $Q(x)$ is non-rewinding, it uses the input x exclusively in a fixed position i, together with a fixed prefix $(x_1, \dots, x_{i-1})$, in all calls to $C^{(k)}$. We first extract i and the prefix.

We now explain a crucial point: how to interact with A_W in order to cheat. We simulate the $i - 1$ interactions of A_W with B_S^* up to and including round $i - 1$ using $(x_1, \dots, x_{i-1})$ as the input bit and randomness of A. In round i, we continue with the *actual* interaction with A_W. Here, A_W uses randomness x (on which we, however, do not have access).

After this interaction, we need to be able to extract the bit c of A_W. For this, we evaluate $Q(x)$, which we claim is possible. Since Q is oblivious and deterministic, the only difficulty is in evaluating the calls to $C^{(k)}(x_1, b_1, \dots, x_k, b_k, r)$.

All calls use the same values for $x_1, \ldots, x_i$. Recalling how $C^{(k)}$ is defined, we see that we can continue from the state we had after the interaction with A_W in order to evaluate $C^{(k)}$ completely (note that all the b_i are given, so the we can also evaluate the information theoretic protocol I).

We get from Theorem 2 that Q satisfies, almost surely, infinitely often, using (7)

$$\Pr_{x \leftarrow \{0,1\}^n}[Q(x) = P(x)] \geq 1 - \frac{\delta}{2} + \frac{1}{48kn^c} \ . \tag{9}$$

This therefore gives a contradiction to (4): in order to get rid of the "almost surely", we just consider the algorithm which first runs Gen and then applies the above protocol – this only loses a negligible additive term in the probability.

4 Weakly Verifiable Puzzles

4.1 Interactive Weakly Verifiable Puzzles

Consider a bit commitment protocol, in which a sender commits to a single bit b. In a first phase the sender and the receiver enact an interactive protocol, after which the sender holds some opening information y, and the receiver has some way of checking whether (y, b) is a valid decommitment. If the protocol is secure, then it is a computationally hard problem for the sender to come up with two strings y_0 and y_1 such that both $(y_0, 0)$ and $(y_1, 1)$ are valid decommitments, in addition, he may not even know the function the receiver will use to validate a decommitment pair,[5] and thus in general there is no way for the sender to recognize a valid pair (y_0, y_1). We abstract this situation in the following definition; in it we can say that the solver produces no output because in the security property all efficient algorithms are considered anyhow.

Definition 4. *An* interactive weakly verifiable puzzle *consists of a protocol (P, S) and is given by two interactive algorithms P and S, in which P (the problem poser) produces as output a circuit Γ, and S (the solver) produces no output.*

The success probability *of an interactive algorithm S^* in solving a weakly verifiable puzzle (P, S) is:*

$$\Pr[y = \langle P, S^* \rangle_{S^*}; \Gamma(y) = 1] \tag{10}$$

The puzzle is non-interactive *if the protocol consists of P sending a single message to S.*

[5] One might want to generalize this by saying that in order to open the commitment, sender and receiver enter yet another interactive protocol. However, our presentation is without loss of generality: the sender can send the randomness he used in the first protocol instead. The receiver then checks, if this randomness together with b indeed produces the communication in the first round, and whether in a simulation of the second protocol he accepts.

Our definition of a non-interactive weakly verifiable puzzle coincides with the usual one [4]. The security property of an interactive weakly verifiable puzzle is that for any algorithm (or circuit) S^* of a restricted class, the success probability of S^* is bounded.

An important property is that S^* does not get access to Γ. Besides bit commitment above, an example of such a puzzle is a CAPTCHA. In both cases it is not obvious whether a given solution is actually a correct solution.

4.2 Strengthening interactive weakly verifiable puzzles

Suppose that g is a monotone boolean function with k bits of input, and $(P^{(1)}, S^{(1)})$ is a puzzle. We can consider the following new puzzle $(P^{(g)}, S^{(g)})$: the sender and the receiver sequentially create k instances of $(P^{(1)}, S^{(1)})$, which yields circuits $\Gamma^{(1)}, \ldots, \Gamma^{(k)}$ for P. Then $P^{(g)}$ outputs the circuit $\Gamma^{(g)}$ which computes $\Gamma^{(g)}(y_1, \ldots, y_k) = g(\Gamma^{(1)}(y_1), \ldots, \Gamma^{(k)}(y_k))$.

Intuitively, if no algorithm solves a single puzzle $(P^{(1)}, S^{(1)})$ with higher probability than δ, the probability that an algorithm solves $(P^{(g)}, S^{(g)})$ should not be more than approximately $\Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1]$. (Recall that μ_δ^k is the distribution on k-bits, where each bit is independent and 1 with probability δ.) The following theorem states exactly this.

Theorem 4. *There exists an algorithm* $\mathrm{Gen}(C, g, \epsilon, \delta, n)$ *which takes as input a circuit C, a monotone function g, and parameters ϵ, δ, n, and produces a circuit D such that the following holds. If C is such that*

$$\Pr[\Gamma^{(g)}(\langle P^{(g)}, C\rangle_C) = 1] \geq \Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] + \epsilon, \tag{11}$$

then, D satisfies almost surely,

$$\Pr[\Gamma^{(1)}(\langle P^{(1)}, D\rangle_D) = 1] \geq \delta + \frac{\epsilon}{6k}. \tag{12}$$

Additionally, Gen *and D only require oracle access to both g and C, and D is non-rewinding.*

Furthermore, $\mathrm{size}(D) \leq \mathrm{size}(C) \cdot \frac{6k}{\epsilon} \log(\frac{6k}{\epsilon})$ *and* Gen *runs in time* $\mathrm{poly}(k, \frac{1}{\epsilon}, n)$ *with oracle calls to C.*

The monotone restriction on g in the previous theorem is necessary. For example, consider $g(b) = 1 - b$. It is possible to satisfy g with probability 1 by producing an incorrect answer, but $\Pr_{u \leftarrow \mu_\delta}[g(u) = 1] = 1 - \delta$.

4.3 Proof of Theorem 4

Algorithm Description. If $k = 1$, Gen creates the circuit D which runs C and outputs its answer. Then either g is the identity or a constant function. If g is the identity, the statement is trivial. If g is a constant function, the statement is vacuously true. D is non-rewinding.

In the general case, we need some notation. For $b \in \{0,1\}$, let $\mathcal{G}_b$ denote the set of inputs $\mathcal{G}_b := \{b_1, \dots, b_k | g(b, b_2, \dots, b_k) = 1\}$ (i.e., the first input bit is disregarded and replaced by b). We remark that $\mathcal{G}_0 \subseteq \mathcal{G}_1$ due to monotonicity of g. We will commonly denote by $u = u_1 u_2 \cdots u_k \in \{0,1\}^k$ an element drawn from μ_δ^k. After a given interaction of C with $P^{(g)}$, let $c = c_1 c_2 \cdots c_k \in \{0,1\}^k$ denote the string where c_i is the output of $\Gamma^{(i)}$ on input y_i, which is the ith output of C. We denote the randomness used by $P^{(g)}$ in execution i by π_i.

For $\pi^*, b \in \{0,1\}^n \times \{0,1\}$ we now define the surplus $S_{\pi^*,b}$. It denotes how much better C performs than "it should", in the case where the randomness of $P^{(g)}$ in the first instance is fixed to π^*, and the output of $\Gamma^{(1)}(y_1)$ is ignored (i.e., we don't care whether C solves the first puzzle right), and b is used instead:

$$S_{\pi^*,b} := \Pr_{\pi^{(k)}}[c \in \mathcal{G}_b | \pi_1 = \pi^*] - \Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_b], \tag{13}$$

where the first probability is also over the interaction between $P^{(g)}$ and C as well as randomness C uses (if any).

The algorithm then works as follows: first pick $\frac{6k}{\epsilon}\log(n)$ candidates π^* for the randomness of $P^{(g)}$ in the first position. For each of those, simulate the interaction $(P^{(g)}, C)$ and then get estimates $\widetilde{S}_{\pi^*,0}$ and $\widetilde{S}_{\pi^*,1}$ of $S_{\pi^*,0}$ and $S_{\pi^*,1}$ such that $|\widetilde{S}_{\pi^*,b} - S_{\pi^*,b}| \le \frac{\epsilon}{4k}$ almost surely.

We consider two cases:

- One of the estimates satisfies $\widetilde{S}_{\pi^*,b} \ge (1 - \frac{3}{4k})\epsilon$.
  In this case, we fix $\pi_1 := \pi^*$ and $c_1 := b$, and invoke $\mathrm{Gen}(C', g', (1-\frac{1}{k})\epsilon, \delta, n)$, using the function $g'(b_2, \dots, b_k) = g(c_1, b_2, \dots, b_k)$ and circuit C' which is defined as follows: C' first (internally) simulates an interaction of $P^{(1)}$ with C, then follows up with an interaction with $P^{(g')}$.
- For all estimates $\widetilde{S}_{x^*,b} < (1 - \frac{3}{4k})\epsilon$.
  In this case, we output the following circuit D^C: in a first phase, use C to interact with $P^{(1)}$. In the second phase, simulate $k-1$ interactions with $P^{(1)}$ and obtain $(y_1, \dots, y_k) = C(x, x_2, \dots, x_k)$. For $i = 2, \dots, k$ set $c_i = \Gamma_i(y_i)$. If $c = (0, c_2, \dots, c_k) \in \mathcal{G}_1 \setminus \mathcal{G}_0$, return y_1, otherwise repeat the second phase $\frac{6k}{\epsilon}\log(\frac{6k}{\epsilon})$ times. If all attempts fail, return the special value $\perp$ (or an arbitrary answer).

Due to space constraints, the proof of correctness of the above algorithm is omitted, but can be found in the full version [22].

5 Example: Bit Commitment

Theorems 3 and 4 can be used to show how to strengthen bit commitment protocols, which was the main open problem in [16]. We explain this as an example here. Assume we have given a weak bit protocol, where a cheating receiver can guess a bit after the commitment phase with probability $1 - \frac{\beta}{2}$, and a cheating sender can change the bit he committed to with probability α. We show that such a protocol can be strengthened if $\alpha < \beta - 1/\operatorname{poly}(n)$.

We should point out that a different way to prove a similar theorem exists: one can first show that such a weak bit-commitment protocol implies one-way functions (using the techniques of [27]). The long sequence of works [17, 32, 36, 33, 14] imply that one-way functions are sufficient to build bit commitment protocols (the first two papers will yield statistically binding protocols, the last three statistically hiding protocols). However, this will be less efficient and also seems less natural than the method we use here.

In the example, we first define weak bit commitment protocols. We then recall a theorem by Valiant [37], and then show how to use it to strengthen bit commitment. However, due to space constraints, the example only appears in the full version [22].

6 Acknowledgments

We would like to thank the anonymous referees for useful comments.

References

1. Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate bregman projections. In *SODA*, pages 1193–1200, 2009.
2. Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *RANDOM-APPROX*, pages 200–215, 2003.
3. Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *FOCS 1997*, pages 374–383, 1997.
4. Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *TCC 2005*, pages 17–33, 2005.
5. Kai-Min Chung and Feng-Hao Liu. Parallel repetition theorems for interactive arguments. In *TCC 2010*, pages 19–36, 2010.
6. Kai-Min Chung, Feng-Hao Liu, Chi-Jen Lu, and Bo-Yin Yang. Efficient string-commitment from weak bit-commitment and full-spectrum theorem for puzzles. Manuscript, 2009.
7. Ivan Dåmgard, Serge Fehr, Kirill Morozov, and Louis Salvail. Unfair noisy channels and oblivious transfer. In Moni Naor, editor, *TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 355–373, 2004.
8. Ivan Dåmgard, Joe Kilian, and Louis Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 56–73, 1999.
9. Yevgeniy Dodis, Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Security amplification for interactive cryptographic primitives. In *TCC 2009*, 2009.
10. Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360, 2004.
11. Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. Technical Report TR95-050, Electronic Colloquium on Computational Complexity (ECCC), 1995.

12. Iftach Haitner. A parallel repetition theorem for any interactive argument. In *FOCS 2009*, pages 241–250, 2009.
13. Iftach Haitner, Danny Harnik, and Omer Reingold. On the power of the randomized iterate. In Cynthia Dwork, editor, *Advances in Cryptology — CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, 2006.
14. Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 1–10, 2007.
15. Iftach Haitner, Omer Reingold, and Salil Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In *Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing*, 2010.
16. Shai Halevi and Tal Rabin. Degradation and amplification of computational hardness. In *TCC 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 626–643, 2008.
17. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
18. Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. An efficient parallel repetition theorem. In *TCC 2010*, pages 1–18, 2010.
19. Thomas Holenstein. Key agreement from weak bit agreement. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 664–673, 2005.
20. Thomas Holenstein. *Strengthening Key Agreement using Hard-Core Sets*. PhD thesis, ETH Zürich, 2006.
21. Thomas Holenstein and Renato Renner. One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In Victor Shoup, editor, *Advances in Cryptology — CRYPTO 2005*, Lecture Notes in Computer Science, 2005.
22. Thomas Holenstein and Grant Schoenebeck. General hardness amplification of predicates and puzzles. *CoRR*, abs/1002.3534, 2010.
23. Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *The 36th Annual Symposium on Foundations of Computer Science*, pages 538–545, 1995.
24. Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *The 47th Annual Symposium on Foundations of Computer Science*, pages 187–196, 2006.
25. Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Chernoff-type direct product theorems. *Journal of Cryptology*, 2009.
26. Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: simplified, optimized, and derandomized. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 579–588, 2008.
27. Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *The 30th Annual Symposium on Foundations of Computer Science*, pages 230–235, 1989.
28. Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.
29. Charanjit S. Jutla. Almost optimal bounds for direct product threshold theorem. In *TCC 2010*, pages 37–51, 2010.

30. Adam R. Klivans and Rocco A. Servedio. Boosting and hard-core sets. In *The 40th Annual Symposium on Foundations of Computer Science*, pages 624–633, 1999.
31. Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.
32. Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.
33. Minh-Huyen Nguyen, Shien Jin Ong, and Salil P. Vadhan. Statistical zero-knowledge arguments for NP from any one-way function. In *The 47th Annual Symposium on Foundations of Computer Science*, pages 3–14, 2006.
34. Rafael Pass and Muthuramakrishnan Venkitasubramaniam. An efficient parallel repetition theorem for arthur-merlin games. In *STOC 2007*, pages 420–429, 2007.
35. Krzysztof Pietrzak and Douglas Wikström. Parallel repetition of computationally sound protocols revisited. In *TCC 2007*, pages 86–102, 2007.
36. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990.
37. Leslie G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5:363–366, 1984.
38. Jürg Wullschleger. Oblivious transfer amplification. In Moni Naor, editor, *Advances in Cryptology — EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 555–572, 2007.
39. Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *The 23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982.