Security Amplification for *Interactive* Cryptographic Primitives

Yevgeniy Dodis[1], Russell Impagliazzo[2], Ragesh Jaiswal[3], and Valentine Kabanets[4]

[1] `dodis@cs.nyu.edu`. New York University.
[2] `russell@cs.ucsd.edu`. University of California at San Diego and IAS.
[3] `rjaiswal@cs.columbia.edu`. Columbia University.
[4] `kabanets@cs.sfu.ca`. Simon Fraser University.

Abstract. Security amplification is an important problem in Cryptography: starting with a "weakly secure" variant of some cryptographic primitive, the goal is to build a "strongly secure" variant of the same primitive. This question has been successfully studied for a variety of important cryptographic primitives, such as one-way functions, collision-resistant hash functions, encryption schemes and weakly verifiable puzzles. However, all these tasks were non-interactive. In this work we study security amplification of *interactive* cryptographic primitives, such as message authentication codes (MACs), digital signatures (SIGs) and pseudorandom functions (PRFs). In particular, we prove direct product theorems for MACs/SIGs and an XOR lemma for PRFs, therefore obtaining nearly optimal security amplification for these primitives.

Our main technical result is a new Chernoff-type theorem for *Dynamic Weakly Verifiable Puzzles*, which we introduce in this paper and which is a generalization of ordinary Weakly Verifiable Puzzles.

1 Introduction

Security amplification is a fundamental cryptographic problem: given a construction C of some primitive P which is only "weakly secure", can one build a "strongly secure" construction C' from C? The first result in this domain is a classical conversion from weak one-way functions to strong one-way function by Yao [Yao82] (see also [Gol01]): if a function f is only mildly hard to invert on a random input x, then, for appropriately chosen n, the function $F(x_1, \ldots, x_n) = (f(x_1), \ldots, f(x_n))$ is very hard to invert. The above result is an example of what is called the *direct product theorem*, which, when true, roughly asserts that simultaneously solving many independent repetitions of a mildly hard task is a much harder "combined task". Since the result of Yao, such direct product theorems have have been successfully used to argue security amplification of several other important cryptographic primitives, such as collision-resistant hash functions [CRS$^+$07], encryption schemes [DNR04] and weakly verifiable puzzles [CHS05,IJK08].

However, all the examples above are non-interactive: namely, after receiving its challenge, the attacker needs to break the corresponding primitives without any further help or interaction. This restriction turns out to be important, as security amplification, and, in particular, direct product theorems become much more subtle for interactive primitives. For example, Bellare, Impagliazzo and Naor [BIN97] demonstrated that that parallel repetition does *not*, in general, reduce the soundness error of multi-round (computationally sound) protocols, and this result was further strengthened by Pietrzak and Wikstrom [PW07]. On the positive side, parallel repetition is known to work for the special case of three-round protocols [BIN97] and constant-round public-coin protocols [PV07]. However, considerably less work has been done in the security amplification of more "basic" cryptographic primitives requiring interaction, such as block ciphers, message authentications codes (MACs), digital signatures (SIGs) and pseudorandom functions (PRFs). For example, Luby and Rackoff [LR86] (see also [NR99,Mye99]) showed how to improve the security of a constant number of pseudorandom permutation generators by composition, while Myers [Mye03] showed that a (non-standard) variant of the XOR lemma [Yao82,Lev87,Imp95,GNW95] holds for PRFs. In particular, the known results for the interactive case are either weaker or more specialized than those for the non-interactive case. The difficulty is that, for instance in the case of MACs, the attacker has oracle access to the corresponding "signing" and "verification" oracles, and the existing techniques do not appear to handle such cases.

In this work we study the question of security amplification of MACs, SIGs and PRFs, showing how to convert a corresponding weak primitive into a strong primitive. In brief, we prove a direct product theorem for MACs/SIGs (and even a Chernoff-type theorem to handle MACs/SIGs with *imperfect completeness*), and a (regular) XOR lemma for PRFs. Before describing these results in more details, however, it is useful to introduce our main technical tool for all these cases — a Chernoff-type theorem for what we call *Dynamic Weakly Verifiable Puzzles* (DWVPs) — which is of independent interest.

Dynamic Weakly Verifiable Puzzles. Recall, (non-dynamic) weakly verifiable puzzles (WVPs) were introduced by Canetti, Halevi and Steiner [CHS05] to capture the class of puzzles whose solutions can only be verified efficiently by the party generating the instance of the puzzle. This notion includes, as special cases, most previously mentioned non-interactive primitives, such as one-way functions, collision-resistant hash functions, one-way encryption schemes, CAPTCHAs, etc. To handle also interactive primitives, such as MACs and SIGs (and also be useful later for PRFs), in Section 3 we generalize this notion to that of *dynamic* WVPs (DWVPs) as follows. Just like in WVPs, one samples a pair (x, α) from some distribution $\mathcal{D}$, where α is the secret advice used to verify proposed solutions r to the puzzle x. Unlike WVPs, however, each x actually defines a *set* of related puzzles, indexed by some value $q \in Q$, as opposed to a single puzzle (which corresponds to $|Q| = 1$). An efficient verification algorithm R for the DWVP uses α *and* the puzzle index q to test if a given solution r is correct. An attacker B has oracle access to this verification procedure. Additionally, the attacker has oracle access to the *hint oracle*: given an index q, the hint oracle returns some hint value $H(\alpha, q)$, presumably "helping" the attacker to solve the puzzle q. The attacker wins the DWVP game if it ever causes the verification oracle to succeed on a query $q \in Q$ not previously queried to the hint oracle. As we see, this abstraction clearly includes MACs and SIGs as special cases. It also generalizes ordinary WVPs, corresponding to $|Q| = 1$. We say that the DWVP is δ-hard, if no (appropriately bounded) attacker can win the above game with probability more than $(1 - \delta)$.

Our main technical result is the following (informally stated) Chernoff-type theorem for DWVPs. Given n independently chosen δ-hard DWVPs on some index set Q, the chance of solving more than $(n - (1 - \gamma)\delta n)$ DWVPs — on the *same* value $q \in Q$ and using less than h "hint" queries $q' \neq q$ — is proportional to $h \cdot e^{-\Omega(\gamma^2 \delta n)}$; the exact statement is given in Theorem 4. Notice, the value $0 < \gamma \leq 1$ measures the "slackness parameter". In particular, $\gamma = 1$ corresponds to the direct product theorem where the attacker must solve all n puzzles (on the same q). However, setting $\gamma < 1$ allows to handle the setting where even the "legitimate" users, — who have an advantage over the attacker, like knowing α or being humans, — can also fail to solve the puzzle with some probability slightly less than $(1 - \gamma)\delta$.

This result generalizes the corresponding Chernoff-type theorem of Impagliazzo, Jaiswal and Kabanets [IJK08] for standard, *non-dynamic*, WVPs. However, the new theorem involves a considerably more complicated proof. The extra difficulties are explained in Section 3.1. In essence, in order to amplify security, the attacker B for the single DWVP must typically execute the assumed attacker A for the "threshold" variant several times, before obtaining sufficient "confidence" in the quality of the solutions output by A. In each of these "auxiliary" runs, however, there is a chance that A will ask a hint query for the index q which is equal to the one that A is going to solve in the "actual" run leading to the forgery, making B's forgery value q "old". Thus, a new delicate argument has to be made to argue security in this scenario. At a high level, the argument is somewhat similar to Coron's improved analysis [Cor00] of the full domain hash signature scheme, although the details differ. See Theorem 4 for the details.

Applications to MACs, SIGs and PRFs. Our main technical result above almost immediately implies security amplification for MACs and SIGs, even with imperfect completeness. For completeness, we briefly state the (asymptotic) result for the MAC case. (The case of SIGs and the exact security version for both cases are immediate.) We assume that the reader is familiar with the basic syntax and the standard Chosen Message Attack (CMA) scenario for a MAC, which is given by a tagging algorithm Tag and the verification algorithm Ver. We denote the secret key by s, and allow the tagging algorithm to be probabilistic (but not stateful). Given the security parameter k, we say that the MAC has *completeness error* $\beta = \beta(k)$ and *unforgeability* $\delta = \delta(k)$, where $\beta < \delta$, if for any message m, $\Pr(\mathsf{Ver}(s, m, \mathsf{Tag}(s, m)) = 1) \geq 1 - \beta$, and that no probabilistic polynomial-time attacker B can forge a valid tag for a "fresh" message m with probability greater than $(1 - \delta)$ during the CMA attack.

The MAC Π is said to be *weak* if $\delta(k) - \beta(k) \geq 1/\mathrm{poly}(k)$, for some polynomial poly, and is said to be *strong* if, for sufficiently large k, $\beta(k) \leq \mathrm{negl}(k)$ and

$\delta(k) \geq 1 - \text{negl}(k)$, where $\text{negl}(k)$ is some negligible function of k. Given an integer n and a number $\gamma > 0$, we can define the "threshold direct product" MAC Π^n in the natural way: the key of Π^n consists of n independent keys for the basic MAC, the tag of m contains the concatenation of all n individual tags of m, and the verification accepts an n-tuples of individual tags if at least $(n - (1 - \gamma)\delta n)$ individual tags are correct. Then, a straightforward application of Theorem 4 gives:

Theorem 1. *Assume Π is a weak MAC. Then one can choose $n = \text{poly}(k)$ and $\gamma > 0$ so that Π^n has completeness error $2^{-\Omega(k)}$ and unforgeability $(1 - \text{negl}(k))$. In particular, Π^n is a strong MAC.*

We then use our direct product result for MACs to argue the XOR lemma for the security amplification of PRFs. Namely, in Section 4.2 we show that the XOR of several independent weak PRFs results in a strong PRF (see Section 4.2 for definitions). It is interesting to compare this result with a related XOR lemma for PRFs by Myers [Mye03]. Meyers observed that the natural XOR lemma above cannot hold for δ-pseudorandom PRFs, where $\delta \geq \frac{1}{2}$. In particular, a PRF one of whose output bits is a constant for some input can potentially reach security (almost) 1/2, but can never be amplified by a simple XOR. Because of this counter-example, Meyers had a more complicated XOR lemma for PRFs, where a separate pad was selected for each δ-pseudorandom PRF, and showed that this variant worked for any $\delta < 1$. In this work, we show that Meyers' counter-example is the worst: the simple XOR lemma holds for δ-pseudorandom PRFs, for any $\delta < \frac{1}{2}$.

The PRF result generally follows the usual connection between the direct product theorems and the XOR lemmas first observed by [GNW95], but with a subtlety. First, it is easy to see that it suffices to consider *Boolean* PRFs. For those, we notice that a δ-pseudorandom PRF is also a $(1 - 2\delta)$-unforgeable (Boolean) MAC (this is where $\delta < \frac{1}{2}$ comes in). Then, we apply the direct product theorem to obtain a strong (non-Boolean) MAC. At this stage, one typically applies the Goldreich-Levin [GL89] theorem to argue that the XOR of a random subset of (strong) MACs is a PRF. Unfortunately, as observed by Naor and Reingold [NR98], the standard GL theorem *does not work in general* for converting unpredictability into pseudorandomness, at least when the subset is *public* (which will ultimately happen in our case). However, [NR98] showed that the conversion *does* work when r is kept *secret*. Luckily, by symmetry, it is easy to argue that for "direct product MACs", keeping r secret or public does not make much difference. Indeed, by slightly adjusting the analysis of [NR98] to our setting, we will directly obtain the desired XOR lemma for PRFs.

Finally, in Section 4.1 we observe a simple result regarding the security amplification of pseudorandom generators (PRGs). This result does not use any new techniques (such as our Chernoff-type theorem). However, we state it for completeness, since it naturally leads to the (more complicated) case of PRFs in Section 4.2 and, as far as we know, it has never explicitly appeared in the literature before.

2 Preliminaries

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

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

Theorem 2 ([GL89]). *There is a probabilistic algorithm Dec with the following property. Let $a \in \{0,1\}^k$ be any string, and let $O : \{0,1\}^k \to \{0,1\}$ be any predicate such that $|\mathbf{Pr}_{z \in \{0,1\}^k}[O(z) = \langle a, z \rangle] - 1/2| \geq \nu$ for some $\nu > 0$. Then, given ν and oracle access to the predicate O, the algorithm Dec runs in time $\mathrm{poly}(k, 1/\nu)$, and outputs a list of size $O(1/\nu^2)$ such that, with probability at least $3/4$, the string a is on the list.*

Theorem 3 ([Gol01]). *Let $\{X_n\}_{n \in \mathbb{N}}$ be a distribution ensemble. If $\{X_n\}_{n \in \mathbb{N}}$ is $\delta(n)$-unpredictable, then it is $(n \cdot \delta(n))$-pseudorandom. Also, if $\{X_n\}_{n \in \mathbb{N}}$ is $\delta(n)$-pseudorandom, then it is $\delta(n)$-unpredictable.*

2.1 Samplers

We will consider bipartite graphs $G = G(L \cup R, E)$ defined on a bipartition $L \cup R$ of vertices; we think of L as left vertices, and R as right vertices of the graph G. We allow graphs with multiple edges. For a vertex v of G, we denote by $N_G(v)$ the multiset of its neighbors in G; if the graph G is clear from the context, we will drop the subscript and simply write $N(v)$. Also, for a vertex x of G, we denote by E_x the set of all edges in G that are incident to x. We say that G is *bi-regular* if the degrees of vertices in L are the same, and the degrees of vertices in R are the same.

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

We will use the following properties of samplers (proved in [IJKW08,IJK08]). The first property says that for any two large vertex subsets W and F of a sampler, the fraction of edges between W and F is close to the product of the densities of W and F.

Lemma 2 ([IJKW08]). *Suppose $G = G(L \cup R, E)$ is a λ-sampler. Let $W \subseteq R$ be any set of measure at least τ, and let $V \subseteq L$ be any set of measure at least β. Then, for all $0 < \nu < 1$ and $\lambda_0 = \lambda(\beta, \nu)$, we have $\mathbf{Pr}_{x \in L, y \in N(x)}[x \in V \ \& \ y \in W] \geq \beta(1-\nu)(\tau - \lambda_0)$, where the probability is for the random experiment of first picking a random node $x \in L$ uniformly at random, and then picking a uniformly random neighbor y of x in the graph G.*

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

Lemma 3 ([IJKW08]). *Suppose $G = G(L \cup R, E)$ is a λ-sampler, with the right degree D. Let $W \subseteq R$ be any subset of density at least τ, and let $G' = G(L \cup W, E')$ be the induced subgraph of G (obtained after removing all vertices in $R \setminus W$), with the edge set E'. Let $Col : E' \to \{red, green\}$ be any coloring of the edges of G' such that at most $\eta D|W|$ edges are colored red, for some $0 \le \eta \le 1$. Then, for all $0 < \nu, \beta < 1$ and $\lambda_0 = \lambda(\beta, \nu)$, we have*

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

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

We shall also need a generalization of Lemma 3 for the case of weighted graphs, proved in [IJK08]. Here we consider bipartite graphs $G = G(L \cup R, E)$ whose edges are assigned weights in the interval $[0, 1]$ satisfying the following property: for every right vertex $y \in R$, all the edges incident on y are assigned the same weight. Let $w_0, w_1, \dots$ be the distinct weights of the edges of G, in decreasing order. The vertex set R of such a weighted graph is naturally partitioned into subsets $W_0, W_1, \dots$, where W_i is the subset of all those vertices in R whose incident edges have weight w_i. Intuitively, such a partitioning of R defines a new induced graph G' where a vertex in W_i is present in G' with probability w_i. (In the setting of Lemma 3, there are two sets $W_0 = W$ and $W_1 = R \setminus W$, with $w_0 = 1$ and $w_1 = 0$.)

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

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

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

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

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

Next consider the following experiment. Pick a vertex $x \in L$ uniformly at random, then pick an edge e incident to x with probability proportionate to $wt(e)$

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

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

Lemma 4 ([IJK08]). *Suppose $G = G(L \cup R, E)$ is a λ-sampler with the right degree D. Let $wt : E \to [0, 1]$ be the weight function over the edges of G such that, for each $y \in R$, the weights of the edges $e \in E_y$ incident to y are the same. Let $w_0, w_1, \ldots$ be the distinct weights of the edges of G, in decreasing order, and let $W_0, W_1, \ldots$ be the partitioning of the vertex set R so that each W_i is the subset of all those vertices in R whose incident edges have the weight w_i. Suppose that W_0 has the measure at least τ in the set R.*

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

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

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

3 Dynamic Weakly Verifiable Puzzles

We consider the following generalization of weakly verifiable puzzles (WVP) [CHS05], which we call *dynamic weakly verifiable puzzles (DWVP).*

Definition 1 (Dynamic Weakly Verifiable Puzzle). *A DWVP Π is defined by a distribution $\mathcal{D}$ on pairs of strings (x, α); here α is the advice used to generate and evaluate responses to the puzzle x. Unlike the case of WVP, here the string x defines* a set *of puzzles, (x, q) for $q \in Q$ (for some set Q of indices). There is a probabilistic polynomial-time computable relation R that specifies which answers are solutions for which of these puzzles: $R(\alpha, q, r)$ is true iff response r is correct answer to puzzle q in the collection determined by α. Finally, there is also a probabilistic polynomial-time computable* hint *function $H(\alpha, q)$.*

A solver can make a number of queries: query $hint(q)$ asks for $H(\alpha, q)$, the hint for puzzle number q; a verification query $V(q, r)$ asks whether $R(\alpha, q, r)$. The solver succeeds if it makes an accepting verification query for a q where it has not previously made a hint query on q.

Clearly, WVP is a special case of DWVP when $|Q| = 1$. A MAC is also a special case of DWVP where α is a secret key, x is the empty string, queries q are messages, a hint is to give the MAC of a message, and correctness is for a (valid message, MAC) pair. Signatures are also a special case with α being a secret key, x a public key, and the rest similar to the case of MACs.

We give security amplification for such weakly verifiable puzzle collections, using direct products. First, let us define an n-wise direct product for DWVPs.

Definition 2 (n-wise direct-product of DWVPs). *Given a DWVP Π with $\mathcal{D}$, R, Q, and H, its n-wise direct product is a DWVP Π^n with the product distribution $\mathcal{D}^n$ producing n-tuples $(\alpha_1, x_1), \ldots, (\alpha_n, x_n)$. For a given n-tuple $\bar{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and a query $q \in Q$, the new hint function is $H^n(\bar{\alpha}, q) = (H(\alpha_1, q), \ldots, H(\alpha_n, q))$. For parameters $0 \le \gamma, \delta \le 1$, we say that the new relation $R^k((\alpha_1, \ldots, \alpha_n), q, (r_1, \ldots, r_n))$ evaluates to true if there is a subset $S \subseteq [n]$ of size at least $n - (1-\gamma)\delta n$ such that $\wedge_{i \in S} R(\alpha_i, q, r_i)$.*

A solver of the n-wise DWVP Π^n may ask hint queries $hint^n(q)$, getting $H^n(\bar{\alpha}, q)$ as the answer. A verification query $V^n(q, \bar{r})$ asks if $R^n(\bar{\alpha}, q, \bar{r})$, for an n-tuple $\bar{r} = (r_1, \ldots, r_n)$. We say that the solver succeeds if it makes an accepting verification query for a q where it has not previously made a hint query on q.[1]

Theorem 4 (Security amplification for DWVP (uniform version)). *Suppose a probabilistic t-time algorithm A succeeds in solving the n-wise direct-product of some DWVP Π^n with probability at least ϵ, where $\epsilon \ge (800/\gamma\delta) \cdot (h + v) \cdot e^{-\gamma^2 \delta n/40}$, and h is the number of hint queries* [2]*, and v the number of verification queries made by A. Then there is a uniform probabilistic algorithm B that succeeds in solving the original DWVP Π with probability at least $(1-\delta)$, while making $O((h(h+v)/\epsilon) \cdot \log(1/\gamma\delta))$ hint queries, only one verification query, and having a running time*

$$O\left(((h+v)^4/\epsilon^4) \cdot t + (t + \omega h) \cdot (h+v)/\epsilon \cdot \log(1/\gamma\delta)\right).$$

Here ω denotes the maximum time to generate a hint for a given query. The success probability of B is over the random input puzzle of Π and internal randomness of B.

Note that B in the above theorem is a uniform algorithm. We get a reduction in running time of an algorithm for attacking Π if we allow it to be non-uniform. The algorithm B above (as we will see later in the proof) samples a suitable hash function from a family of pairwise independent hash functions and then uses the selected function in the remaining construction. In the non-uniform version of the above theorem, we can skip this step and assume that the suitable hash function is given to it as advice. Following is the non-uniform version of the above theorem.

[1] We don't allow the solver to make hint queries $(q_1, \ldots, q_n)$, with different q_i's, as this would make the new k-wise DWVP completely insecure. Indeed, the solver could ask cyclic shifts of the query $(q_1, \ldots, q_n)$, and thus learn the answers for q_1 in all n positions, without actually making the hint query $(q_1, \ldots, q_1)$.

[2] Note that when $h = 0$, we're in the case of WVPs.

Theorem 5 (Security amplification for DWVP (non uniform version)). *Suppose a probabilistic t-time algorithm A succeeds in solving the n-wise direct-product of some DWVP Π^n with probability at least ϵ, where $\epsilon \geq (800/\gamma\delta) \cdot (h + v) \cdot e^{-\gamma^2\delta n/40}$, h is the number of hint queries, and v the number of verification queries made by A. Then there is a probabilistic algorithm B that succeeds in solving the original DWVP Π with probability at least $1 - \delta$, while making $O((h \cdot (h+v)/\epsilon)) \cdot \log(1/\gamma\delta))$ hint queries, only one verification query, and having the running time $O\left((t + \omega h) \cdot ((h+v)/\epsilon) \cdot \log\left(1/\gamma\delta\right)\right)$, where ω denotes the maximum time to generate a hint for a given query. The success probability of B is over the random input puzzle of Π and internal randomness of B.*

3.1 Intuition

We want to solve a single instance of DWVP Π, using an algorithm A for the n-wise direct-product Π^n, and having access to the hint-oracle and the verification-oracle for Π. The idea is to "embed" our unknown puzzle into an n-tuple of puzzles, by generating the $n-1$ puzzles at random by ourselves. Then we simulate algorithm A on this n-tuple of puzzles. During this simulation, we can answer the hint queries made by A by computing the hint function on our own puzzles and by making the appropriate hint query to the hint-oracle for Π. We will answer all verification queries of A by 0 (meaning "failure"). At the end, we see if A made a verification query which was "sufficiently" correct in the positions corresponding to our own puzzles; if so, we make a probabilistic decision to output this query (for the position of our unknown input puzzle).

To decide whether to believe or not to believe the verification query made by A, we count the number of correct answers it gave for the puzzles we ourselves generated (and hence can verify), and then believe with probability inverse-exponentially related to the number of incorrect answers we see (i.e., the more incorrect answers we see, the less likely we are to believe that A's verification query is correct for the unknown puzzle); since we allow up to $(1-\gamma)\delta n$ incorrect answers, we will discount these many incorrect answers, when making our probabilistic decision.

Such a "soft" decision algorithm for testing if an n-tuple is good has been proposed in [IW97], and later used in [BIN97,IJK08]. Using the machinery of [IJK08], we may assume, for the sake of intuition, that we can decide if a given verification query $(q, \bar{r})$ made by A is correct (i.e., is correct for at least $n - (1-\gamma)\delta n$ of r_i's in the n-tuple $\bar{r}$).

Since A is assumed to succeed on at least ϵ fraction of n-tuples of puzzles, we get from A a correct verification query with probability at least ϵ (for a random unknown puzzle, and random $n-1$ self-generated puzzles). Hence, we will produce a correct solution to the input puzzle of Π with probability at least ϵ.

To improve this probability, we would like to repeatedly sample $n-1$ random puzzles, simulate A on the obtained n-tuple of puzzles (including the input puzzle in a random position), and check if A produces a correct verification query. If we repeat for $O(\log 1/\delta)/\epsilon)$ iterations, we should increase our success probability for solving Π to $1-\delta$.

However, there is a problem with such repeated simulations of A on different n-tuples of puzzles: in some of its later runs, A may make a successful verification query for the same q for which it made a hint query in an earlier run. Thus, we need to make sure that a successful verification query should not be one of the hint queries asked by A in one of its previous runs. We achieve this by randomly partitioning the query space Q into the "attack" queries P, and "hint" queries. Here P is a random variable such that any query has probability $\frac{1}{2(h+v)}$ of falling inside P. We will define the set P by picking a random hash function *hash* from Q to $\{0, 1, \ldots, 2(h+v)-1\}$, and setting $P = P_{hash}$ to be the preimages of 0 of *hash*.

We say that the *first success query* for A is the first query where a successful verification query without a previous hint query is made. A *canonical success* for attacker A with respect to P is an attack so that the first successful verification query is in P and all earlier queries (hint or verification) are not in P.

We will show that the expected fraction of canonical successes for P_{hash} is at least $\frac{\epsilon}{4(h+v)}$. We will also give an efficient algorithm (the **Pick-hash** procedure below) that finds a hash function *hash* so that the fraction of canonical successes for P_{hash} is close to the expected fraction. Then we test random candidates for being canonical successes with respect to P_{hash}.

Due to this extra complication (having to separate hint and verification queries), we lose on our success probability by a factor of $8(h+v)$ compared to the case of WVPs analyzed in [IJK08]. The formal proof of the theorem is given in the following subsection.

3.2 Proof of Theorem 4 and 5

For any mapping $hash : Q \to \{0, ..., 2(h+v)-1\}$, let P_{hash} denote the preimages of 0. Also, as defined in the previous section, a canonical success for an attacker A with respect to $P \subseteq Q$ is an attack so that the first successful verification query is in P and all earlier queries (hint or verification) queries are not in P. The proof of Theorems 4 and 5 follows from the following two lemmas.

Lemma 5. *Let A be an algorithm which succeeds in solving the n-wise direct-product of some DWVP Π^n with probability at least ϵ while making h hint queries, v verification queries and have a running time t. Then there is a probabilistic algorithm which runs in time $O(((h+v)^4/\epsilon^4)\cdot t)$ and with high probability outputs a function $hash : Q \to \{0, ..., 2(h+v)-1\}$ such that the canonical success probability of A with respect to the set P_{hash} is at least $\frac{\epsilon}{8(h+v)}$.*

Lemma 6. *Let $hash : Q \to \{0, ..., 2(h+v)-1\}$ be a function. Let A be an algorithm such that the canonical success probability of A over an n-wise DWVP Π^n with respect to P_{hash} is at least $\epsilon' = (100/\gamma\delta)\cdot e^{-\gamma^2\delta n/40}$. Furthermore, let A makes h hint queries and v verification queries and have a running time*

t. Then there is a probabilistic algorithm B that succeeds in solving the original DWVP Π with probability at least $1 - \delta$, while making $O((h(h+v)/\epsilon) \cdot \log(1/\gamma\delta))$ hint queries, only one verification query, and having the running time $O\left((t + \omega h) \cdot (h/\epsilon) \cdot \log\left(1/\gamma\delta\right)\right)$, where ω denotes the maximum time to generate a hint for a given query.

In the remaining subsection, we give the proof of Lemmas 5 and 6. Note that the proof of Lemma 6 is very similar to the analysis of WVPs in [IJK08].

Proof (proof of Lemma 5). Let $\mathcal{H}$ be a pairwise independent family of hash functions mapping Q into $\{0, ..., (2(h+v)-1)\}$. First note that for a randomly chosen function $hash \xleftarrow{\$} \mathcal{H}$, P_{hash} is a random variable denoting the partition of Q into two parts which satisfies the following properties:

$$\forall q_1, q_2 \in Q, \quad \mathbf{Pr}[q_1 \in P_{hash} \mid q_2 \in P_{hash}] = \mathbf{Pr}[q_1 \in P_{hash}] = \frac{1}{2(h+v)} \tag{1}$$

For any fixed choice of $\bar{\alpha} = (\alpha_1, ..., \alpha_n)$, let $q_1^{\bar{\alpha}}, ..., q_h^{\bar{\alpha}}$ denote the hint queries of A and let $(q_{h+1}^{\bar{\alpha}}, \bar{r}_{h+1}^{\bar{\alpha}}), ..., (q_{h+v}^{\bar{\alpha}}, \bar{r}_{h+v}^{\bar{\alpha}})$ denote the verification queries of A. Also, let $(q_j^{\bar{\alpha}}, \bar{r}_j^{\bar{\alpha}})$ denote the first successful verification query, in the case A succeeds, and let it denote any arbitrary verification query in the case A fails. Furthermore, let $E_{\bar{\alpha}}$ denote the event that $q_1^{\bar{\alpha}}, ..., q_h^{\bar{\alpha}}, q_{h+1}^{\bar{\alpha}}, ..., q_{j-1}^{\bar{\alpha}} \notin P_{hash}$ and $q_j^{\bar{\alpha}} \in P_{hash}$. We bound the probability of the event $E_{\bar{\alpha}}$ as follows.

Lemma 7. *For each fixed $\bar{\alpha}$, we have $\mathbf{Pr}_{P_{hash}}[E_{\bar{\alpha}}] \geq \frac{1}{4(h+v)}$.*

Proof. We have

$$\begin{aligned}\mathbf{Pr}_{P_{hash}}[E_{\bar{\alpha}}] &= \mathbf{Pr}[q_j^{\bar{\alpha}} \in P_{hash} \ \&\ \forall i < j, q_i^{\bar{\alpha}} \notin P_{hash}] \\ &= \mathbf{Pr}[q_j^{\bar{\alpha}} \in P_{hash}] \cdot \mathbf{Pr}[\forall i < j, q_i^{\bar{\alpha}} \notin P_{hash} \ \mid \ q_j^{\bar{\alpha}} \in P_{hash}].\end{aligned}$$

By (1), we get that the latter expression is equal to $\frac{1}{2(h+v)} \cdot (1 - \mathbf{Pr}[\exists i < j, q_i^{\bar{\alpha}} \in P_{hash} \ \mid \ q_j^{\bar{\alpha}} \in P_{hash}])$, which, by the union bound, is at least

$$\frac{1}{2(h+v)} \cdot \left(1 - \sum_{i<j} \mathbf{Pr}[q_i^{\bar{\alpha}} \in P_{hash} \ \mid \ q_j^{\bar{\alpha}} \in P_{hash}]\right).$$

Finally, using the pairwise independence property (1), we conclude that

$$\mathbf{Pr}[E_{\bar{\alpha}}] \geq \frac{1}{2(h+v)} \cdot \left(1 - \sum_{i<j} \mathbf{Pr}[q_i^{\bar{\alpha}} \in P_{hash}]\right) \geq \frac{1}{4(h+v)},$$

as required. □

Let T denote the "good" set corresponding to A's attack, that is, $T = \{\bar{\alpha} : \ A's\ attack\ succeeds\}$. We have $Pr[\bar{\alpha} \in T] \geq \epsilon$. Consider the following random variable: $G_{P_{hash}} = \{\bar{\alpha} \mid \bar{\alpha} \in T \text{ and } E_{\bar{\alpha}}\}$. So, $G_{P_{hash}}$ contains those $\bar{\alpha}$'s for which A has canonical success.

Since $\forall \bar{\alpha} \in G, \mathbf{Pr}_{P_{hash}}[E_{\bar{\alpha}}] \geq \frac{1}{4(h+v)}$, using linearity of expectation we get $\mathbf{Exp}_{P_{hash}}[\mathbf{Pr}_{\bar{\alpha}}[\bar{\alpha} \in G_{P_{hash}}]] \geq \frac{\epsilon}{4(h+v)}$. Hence, by averaging, we get that with probability at least $\frac{\epsilon}{8(h+v)}$ over the randomness of P_{hash}, there is at least $\frac{\epsilon}{8(h+v)}$ chance that a randomly chosen $\bar{\alpha} \in G_{P_{hash}}$. Let us call such P_{hash}'s "good". The subroutine **Pick-hash** (see figure 2) uses the natural sampling technique and runs in time $O(((h+v)^4/\epsilon^4) \cdot t)$ to return a mapping *hash* such that P_{hash} is good.

Pick-hash

00. Let $\mathcal{H}$ be a pairwise independent family of hash functions which maps Q into $\{0, 1, ..., (2h-1)\}$.
01. Repeat lines $(2-15)$ at most $64(h+v)^2/\epsilon^2$ times:
02. $\quad hash \xleftarrow{\$} \mathcal{H}$
03. $\quad$ Let P_{hash} denote the subset of all queries q such that $hash(q) = 0$
04. $\quad count \leftarrow 0$
05. $\quad$ Repeat for at most $64(h+v)^2/\epsilon^2$ times:
06. $\quad\quad$ Pick $\bar{\alpha} = (\alpha_1, ..., \alpha_n)$ randomly
07. $\quad\quad$ Execute A
08. $\quad\quad\quad$ When A asks a hint query q
09. $\quad\quad\quad\quad$ **If** $(q \in P_{hash})$, then abort A and continue with step 5
10. $\quad\quad\quad\quad$ Let $(r_1, ..., r_n)$ be hints to query q for puzzle sets $x_1, ..., x_n$
11. $\quad\quad\quad\quad$ return $(r_1, ..., r_n)$ to A
12. $\quad\quad\quad$ When A asks a verification query $(q, \bar{r})$
13. $\quad\quad\quad\quad$ **If** $(R^n(\bar{\alpha}, q, \bar{r}) = 1)$ and $q \in P_{hash}$ then
14. $\quad\quad\quad\quad\quad\quad$ increase *count* by 1 and continue at step 5
15. $\quad$ **If** $(count \geq 4(h+v)/\epsilon)$ then return *hash*

Figure 1: Algorithm for picking a *good* hash function

□

Proof (Proof of Lemma 6).

Figure (2) gives the formal description of the algorithm B which uses the algorithm A.

For any fixed $\bar{\alpha}$, let $q_1^{\bar{\alpha}}, ..., q_h^{\bar{\alpha}}$ denote the hint queries made by A and $(q_{h+1}^{\bar{\alpha}}, r_{h+1}^{\bar{\alpha}}), ..., (q_{h+v}^{\bar{\alpha}}, r_{h+v}^{\bar{\alpha}})$ denote the verification queries. Let $j \in [h+v]$ be the first query such that $q_j^{\bar{\alpha}} \in P_{hash}$. For all simulations of A, B correctly answers every hint query of A by itself making a hint query with respect to the DWVP Π which it is trying to solve (lines 10–13). Note that for a single simulation of A (lines 6–25), the simulation is aborted if $j \leq h$ (line 9). Otherwise, B makes a "soft" decision using $(q_j^{\bar{\alpha}}, \bar{r}_j^{\bar{\alpha}})$ to produce its verification query (lines 19–24). Lemma 5 tells that there are at least ϵ' fraction of $\bar{\alpha}$'s such that $(q_j^{\bar{\alpha}}, \bar{r}_j^{\bar{\alpha}})$ is a correct verification query for A(these are $\bar{\alpha}$'s on which A has canonical success). So, intuitively there is a fair chance that B produces a correct verification query.

B(x)
00. Let $\rho = (1 - \gamma/10)$ and $\Theta = (1 - \gamma)\delta n$
01. Let *hash* denote the function as in Lemma 5. For Theorem 4, *hash* is chosen using the subroutine **Pick-hash**. For Theorem 5, we can assume that *hash* is given as advice.
02. Let P_{hash} denote the subset of all queries q such that $hash(q) = 0$
03. Repeat lines $(4 - 23)$ for at most $timeout = O(((h + v)/\epsilon) \cdot \log(1/\gamma\delta))$ steps:
04. Pick $i \xleftarrow{\$} [1..n]$
05. Pick $(n - 1)$ α's randomly. Let $(\alpha_1, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n)$ denote these α's and let $(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$ denote the puzzle sets corresponding to these α's.
06. $S_v \leftarrow \emptyset$
07. Execute $A(x_1, ..., x_{i-1}, x, x_{i+1}, ..., x_n)$
08. When A asks its hint query q
09. If $q \in P_{hash}$ then return $\perp$ to A and halt the current simulation of A
10. B makes a hint query q to get the answer r
11. Let $(r_1, ..., r_{i-1}, r_{i+1}, ..., r_{n-1})$ be the hints for query q for puzzle sets $(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$
12. $\bar{r} \leftarrow (r_1, .., r_{i-1}, r, r_{i+1} .., r_n)$
13. return $\bar{r}$ to A
14. When A asks a verification query $(q, \bar{r})$
15. **If** $q \notin P_{hash}$ then return 0 to A
16. **else**
17. Parse $\bar{r}$ as $(r_1, \ldots, r_n)$
18. $m \leftarrow |\{j : R(\alpha_j, q, r_j) = 1, j \neq i\}|$
19. **If** $(m \geq n - \Theta)$ then
20. with probability 1, B makes a verification query (q, r_i) and halts
21. **else**
22. with probability $\rho^{m-\Theta}$, B makes a verification query (q, r_i) and halts
23. Halt the current simulation of A and continue at line (03)
25. return $(\perp, \perp)$

Figure 2: Algorithm for solving Π

Let *Good* denote the set of $\bar{\alpha}$'s on which A succeed canonically. From the assumption of the Lemma we know that *Good* contains at least ϵ' fraction of $\bar{\alpha}$'s. The remaining task is to argue that this is sufficient to show that B succeeds with high probability. The rest of the analysis, apart from minor details, is similar to the proof of the Direct Product theorem for WVPs from [IJK08], essentially arguing that the lines 17–22 of the algorithm B act as a decision procedure for the set *Good*. Next we give details of these arguments.

Consider the following bipartite graph $G = G(L \cup R, E)$: the set of left vertices L is the set of α's; the right vertices R are all n-tuples $\bar{\alpha} = (\alpha_1, ..., \alpha_n)$; for every $y = (u_1, \ldots, u_n) \in R$, there are n edges $(y, u_1), \ldots, (y, u_n) \in E$. Using Lemma 1, we see that this graph is a λ-sampler for $\lambda(\mu, \nu) = e^{-\nu^2 \mu k/2}$.

For an unknown secret key α, let $\bar{\alpha} = (\alpha_1, \ldots, \alpha_{i-1}, \alpha, \alpha_i, \ldots, \alpha_{n-1})$ be the n-tuple of secret keys that corresponds to the n-tuple of puzzles $(x_1, \ldots, x_n)$ that B will feed to A in line 7. Let $q_1^{\bar{\alpha}}, ..., q_h^{\bar{\alpha}}$ be the A's hint queries and $(q_{h+1}^{\bar{\alpha}}, \bar{r}_{h+1}^{\bar{\alpha}}), ...,$ $(q_{h+v}^{\bar{\alpha}}, \bar{r}_{h+v}^{\bar{\alpha}})$ be the verification queries. Let $j \in [h + v]$ be the first index such that $q_j^{\bar{\alpha}} \in P_{hash}$. Let $(q^{\bar{\alpha}}, \bar{r}^{\bar{\alpha}})$ denote $(q_j^{\bar{\alpha}}, \bar{r}_j^{\bar{\alpha}})$ in case $j > h$ and $(\perp, \perp)$ otherwise.

In the case $(q^{\bar{\alpha}}, \bar{r}^{\bar{\alpha}}) \neq (\perp, \perp)$, B makes a probabilistic decision about using $(q^{\bar{\alpha}}, \bar{r}^{\bar{\alpha}})$ to produce its verification queries. It does that by verifying the answers to the query at all positions other than position i where the unknown α has been planted. Let $(q^{\bar{\alpha}}, r^{\bar{\alpha}})$ denote the verification query made by the algorithm B in this simulation of A. If no verification query is made or if $(q^{\bar{\alpha}}, \bar{r}^{\bar{\alpha}}) = (\perp, \perp)$, then $(q^{\bar{\alpha}}, r^{\bar{\alpha}}) = (\perp, \perp)$. Let (q^B, r^B) denote the single verification query made by B.

First we bound the probability of timeout of B or in other words the probability that $(q^{\bar{\alpha}}, r^{\bar{\alpha}}) = (\perp, \perp)$ in all iterations of B. If $\bar{\alpha} \in Good$, then lines 7–23 will return a verification query with probability 1. Hence, the probability of timeout is at most the probability that B never samples a neighbor $\bar{\alpha} \in Good$ of α in the graph G.

Consider the set H of all those left vertices α of G such that α has less than $\epsilon'/4$ fraction of its neighbors falling into the set *Good*. These are precisely those α's for which B is likely to time out. The next lemma shows that the set H is small.

Lemma 8. *The set H has density at most $\gamma\delta/5$.*

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

Lemma 9. *For every $\alpha \notin H$, we have $\mathbf{Pr}[B \text{ timeouts}\,] \leq \gamma\delta/20$, where the probability is over the internal randomness of B.*

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

Next, we need to show that the probability of $R(\alpha, q^B, r^B) = 0$ conditioned on the event that $(q^B, r^B) \neq (\perp, \perp)$, is small. Note that this conditional probability remains the same across all the simulations of A in lines 4–23. Consider any fixed simulation of A (lines 8–23) such that $(q^{\bar{\alpha}}, \bar{r}^{\bar{\alpha}}) \neq (\perp, \perp)$. Let err be the number of incorrect answers in $\bar{r}^{\bar{\alpha}}$ for the query $q^{\bar{\alpha}}$. Then if $err \leq (1-\gamma)\delta n$, then lines $7-25$ of B produces a verification query with probability 1. Otherwise a verification query is produced with probability that decreases exponentially (by a factor of ρ) as err increases.

For each edge $((\alpha_1, \ldots, \alpha_n), \alpha_i)$ of the graph G, we color this edge green if the i^{th} answer in $\bar{r}^{\bar{\alpha}}$ is the correct for the query $q^{\bar{\alpha}}$, and we color it red otherwise.

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

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

For each α, we have

$$\begin{aligned}&\mathbf{Pr}[R(\alpha, q^B, r^B) = 0 \mid (q^B, r^B) \neq (\perp, \perp)] = \\ &\quad \mathbf{Pr}[\mathcal{E} \text{ outputs red edge incident to } \alpha \mid \mathcal{E} \text{ outputs some edge incident to } \alpha] \end{aligned} \tag{2}$$

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

Rather than analyzing the experiment $\mathcal{E}$, however, we will consider another experiment that is the same as $\mathcal{E}$ except that err is defined as the total number of incorrect answers of $q^{\bar{\alpha}}$ in $\bar{r}^{\bar{\alpha}}$ in *all* positions (i.e., including the position i). That is, we consider the following experiment $\mathcal{E}'$:

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

The following lemma is immediate.

Lemma 10. *For each edge e of G, we have*

$$\mathbf{Pr}[\mathcal{E}' \text{ outputs } e] \leq \mathbf{Pr}[\mathcal{E} \text{ outputs } e] \leq \frac{1}{\rho} \cdot \mathbf{Pr}[\mathcal{E}' \text{ outputs } e].$$

As a corollary of Lemma 10, and by the definition of conditional probability, we get the following.

Lemma 11. *For each α, we have that*

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

Finally, we prove the following.

Lemma 12. $\mathbf{Pr}_\alpha[R(\alpha, q^B, r^B) = 0 \mid (q^B, r^B) \neq (\perp, \perp)] \leq \delta - \gamma\delta/4$.

Proof. By (2) and Corollary 11, we get that $\mathbf{Pr}_\alpha[R(\alpha, q^B, r^B) = 0 \mid (q^B, r^B) \neq (\perp, \perp)]$ is at most

$$\frac{1}{\rho} \cdot \mathbf{Exp}_{\alpha \in L}[\mathbf{Pr}[\mathcal{E}' \text{outputs red edge incident to } \alpha \mid \mathcal{E}' \text{ outputs some edge incident to } \alpha]]. \tag{3}$$

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

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

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

where E_α denotes the set of all edges incident to α. The expectation in (3) is exactly $\mu = \mathbf{Exp}_{\alpha \in L}[\xi(\alpha)]$.

Let Red be the set of all red edges in G, and let $\eta = (1/n|R|)\sum_{e \in Red} wt(e)$. Let us partition the set of $\bar{\alpha}$'s into the subsets $Good_i$, for $i \geq 0$, where $Good_0 = Good$, and for each $i \geq 1$, $Good_i$ contains all those $\bar{\alpha} \in R$ such that there are exactly $((1-\gamma)\delta n + i)$ incorrect answers.

Apply Lemma 4 to G, the partitioning $R = \cup_{i \geq 0} Good_i$ with the corresponding weights $\rho^0, \rho^1, \ldots$, and the measure τ of $Good_0$ being at least ϵ'. We get that μ is at most the maximum of β and the following expression:

$$\frac{|R|\eta}{(1-\nu)(1-\lambda_0/\epsilon')\sum_{i \geq 0}|Good_i|\rho^i}, \tag{4}$$

where $\lambda_0 = \lambda(\beta, \nu)$.

By the definition of the sets $Good_i$, we get that $\eta \leq (1/n|R|)\sum_{i \geq 0}|Good_i|((1-\gamma)\delta n + i)\rho^i$. Using this bound on η, we can upperbound the expression in (4) by

$$\frac{\sum_{i \geq 0}|Good_i|((1-\gamma)\delta + (i/n))\rho^i}{(1-\nu)(1-\lambda_0/\epsilon')\sum_{i \geq 0}|Good_i|\rho^i}. \tag{5}$$

For $t = \gamma\delta n/4$, let us split the sum in the numerator of (5) into two sums: for $0 \leq i \leq t$ and for $i > t$. We can bound each of these two sums as follows:

$$\begin{aligned}\sum_{i=0}^{t}|Good_i|((1-\gamma)\delta + (i/n))\rho^i &\leq ((1-\gamma)\delta + t/n)\sum_{i=0}^{t}|Good_i|\rho^i \\ &\leq (1-3\gamma/4)\delta\sum_{i \geq 0}|Good_i|\rho^i,\end{aligned}$$

and $\sum_{i>t} |Good_i|((1-\gamma)\delta + (i/n))\rho^i \leq \rho^t|R|$. Plugging these bounds into (5), and recalling that $|Good_0| \geq \epsilon'|R|$, we upperbound (5) by

$$\frac{1}{(1-\nu)(1-\lambda_0/\epsilon')}\left((1-3\gamma/4)\delta + \frac{\rho^t|R|}{\sum_{i\geq 0}|Good_i|\rho^i}\right) \leq \frac{(1-3\gamma/4)\delta + \rho^t/\epsilon'}{(1-\nu)(1-\lambda_0/\epsilon')}.$$

Finally, by (3), we get

$$\mathbf{Pr}_\alpha[R(\alpha, q^B, r^B) = 0 \mid (q^B, r^B) \neq (\perp,\perp)] \leq \max\left\{\frac{(1-3\gamma/4)\delta + \rho^t/\epsilon'}{\rho(1-\nu)(1-\lambda_0/\epsilon')}, \frac{\beta}{\rho}\right\},$$

which is at most $\delta - \gamma\delta/4$, for $\rho = 1-\gamma/10$, $\beta = (27/40)\delta$, $\nu = (3/10)\gamma$, $\lambda_0/\epsilon' \leq \gamma/20$ and $\rho^t/\epsilon' \leq \gamma\delta/100$. □

Now we finish the proof of Lemma 6. Let V denote the set of all α's. We have

$$\begin{aligned}\mathbf{Pr}_\alpha[R(\alpha, q^B, r^B)) = 0] = &\frac{1}{|V|}\sum_{\alpha\in H}\mathbf{Pr}[R(\alpha, q^B, r^B) = 0] \\ &+ \frac{1}{|V|}\sum_{\alpha\notin H}\mathbf{Pr}[R(\alpha, q^B, r^B) = 0]. \qquad (6)\end{aligned}$$

The first term on the right-hand side of (6) is at most $\gamma\delta/5$ by Lemma 8. For the second term, we upperbound $\mathbf{Pr}[R(\alpha, q^B, r^B) = 0]$ by $\mathbf{Pr}[R(\alpha, q^B, r^B) = 0 \mid (q^B, r^B) \neq (\perp,\perp)] + \mathbf{Pr}[(q^B, r^B) = (\perp,\perp)]$. We know by Lemma 9 that, for each $\alpha \notin H$, $\mathbf{Pr}[(q^B, r^B) = (\perp,\perp)] \leq \gamma\delta/20$. Thus we get that $\mathbf{Pr}_\alpha[R(\alpha, q^B, r^B) = 0]$ is at most

$$\begin{aligned}&\gamma\delta/5 + \gamma\delta/20 + \frac{1}{|V|}\sum_{\alpha\notin H}\mathbf{Pr}[R(\alpha, q^B, r^B) = 0 \mid (q^B, r^B) \neq (\perp,\perp)] \\ &\leq \gamma\delta/4 + \mathbf{Pr}_\alpha[R(\alpha, q^B, r^B) = 0 \mid (q^B, r^B) \neq (\perp,\perp)],\end{aligned}$$

which is at most δ by Lemma 12.

The bound on the running time and number of queries follows from the description of the algorithm B. □

4 XOR Lemmas for PRGs and PRFs

In this section, we show how to amplify security of pseudorandom (function) generators, using Direct Products (old and new) and the Goldreich-Levin decoding algorithm from Theorem 2.

4.1 Amplifying PRGs

We start with looking at the definition of PRGs.

Definition 3 (PRGs). *Let* $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$ *be a polynomial-time computable generator, stretching* k*-bit seeds to* $\ell(k)$*-bit strings, for* $\ell(k) > k$*, such that* G *is* $\delta(k)$-pseudorandom*. That is, for any probabilistic polynomial-time algorithm* A*, and all sufficiently large* k*, we have* $|\mathbf{Pr}_s[A(G(s)) = 1] - \mathbf{Pr}_x[A(x) = 1]| \leq \delta(k)$*, where* s *is chosen uniformly at random from* $\{0,1\}^k$*, and* x *from* $\{0,1\}^{\ell(k)}$.

Definition 4 (Weak and Strong PRGs). *We say that a PRG* G *is* weak *if it is* δ*-pseudorandom for a constant* $\delta < 1/2$*. We say that a PRG* G *is* strong *if it is* $\delta(k)$*-pseudorandom for* $\delta(k) < 1/k^c$ *for any constant* $c > 0$ *(i.e., negligible).*

For the rest of this subsection, let $n > \omega(\log k)$ and let $n' = 2n$. We show that any weak PRG G_{weak} of stretch $\ell(k) > kn$ can be transformed into a strong PRG G_{strong} as follows:

> $\mathbf{G_{strong}}$: "The seed to G_{strong} is a n-tuple of seeds to G_{weak}, and the output of $G_{strong}(s_1, \ldots, s_n)$ is the bit-wise XOR of the n strings $G_{weak}(s_1), \ldots, G_{weak}(s_n)$."

Theorem 6 (Security amplification for PRGs). *If* G_{weak} *is a weak PRG with stretch* $\ell(k) > kn$*, then the generator* G_{strong} *defined above is a strong PRG, mapping* nk*-bit seeds into* $\ell(k)$*-bit strings.*

Let G_{weak} be δ-pseudorandom for $\delta < 1/2$. We will need the following lemmas for the proof of the above theorem.

Lemma 13 (pseudorandomness implies unpredictability). *Let* $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$ *be a generator which is* δ*-pseudorandom for some* $\delta < 1/2$*. Then for any probabilistic-polynomial time algorithm, and all sufficiently large* k*, the probability that* A *computes the* $(i+1)th$ *bit of* $G(s)$ *(for random* s*) given the first* i *bits of* $G(s)$*, is at most* $(1/2 + \delta)$.

Proof. The proof follows from Theorem 3. □

Lemma 14 (bit-wise direct product lemma). *Let* $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$ *be a pseudorandom generator such that for any probabilistic* t*-time algorithm* A *we have for all* $i \leq \ell(k)$*:*

$$\mathbf{Pr}\left[A(1^k, G(s)[1, ..., (i-1)]) = G(s)[i]\right] \leq 1/2 + \delta.$$

Then for any probabilistic t'*-time algorithm* A' *we have for all* $i \leq \ell(k)$*:*

$$\mathbf{Pr}\left[A'(1^{kn'}, G(s_1)[1, ..., (i-1)], ..., G(s_{n'})[1, ..., (i-1)]) = G(s_1)[i], ..., G(s_{n'})[i]\right] \leq \epsilon.$$

Here $\epsilon = e^{-(\delta n')/c_0}$ *and* $t' = t/poly(\epsilon)$*, for some constant* c_0*.*

Proof. This follows from the direct product theorem for weakly verifiable puzzles (Theorem 1.1, [IJK08]). We model the above problem as a weakly verifiable puzzle as follows: the secret string α for the WVP is the seed s for the pseudorandom generator, the puzzle x is the string $G(s)[1, ...(i-1)]$ and the relation R is defined as $R(s, G(s)[1, ..., (i-1)], y) = 1$ iff $G(s)[i] = y$. □

Lemma 15 (direct product theorem implies xor lemma). *Let $G : \{0,1\}^k \to \{0,1\}^{\ell(k)}$ be a pseudorandom generator such that for any probabilistic t-time algorithm A we have for all $i < \ell(k)$:*

$$\mathbf{Pr}\left[A(1^k, G(s)[1, ..., (i-1)]) = G(s)[i]\right] \leq 1/2 + \delta.$$

Then for any probabilistic t'-time algorithm A' we have for all $i < \ell(k)$:

$$\mathbf{Pr}\left[A'(1^{nk}, G(s_1)[1, ..., (i-1)], ..., G(s_n)[1, ..., (i-1)]) = G(s_1)[i] \oplus ... \oplus G(s_n)[i]\right] \\ \leq 1/2 + \epsilon. \tag{7}$$

Here $\epsilon = e^{-(\delta n)/c_1}$ and $t' = t/poly(\epsilon)$, for some constant c_1.

Proof. For the sake of contradiction, assume that there is an algorithm A' such that (7) does not hold. Consider the following algorithm A'' for computing

$$\langle (G(s_1), ..., G(s_{n'})), r \rangle$$

given a randomly chosen $r \in \{0,1\}^{n'}$ and $G(s_1)[1, ..., i-1], ..., G(s_{n'})[1, ..., i-1]$.

> A'': Given inputs r and $G(s_1)[1, ..., i-1], ..., G(s_{n'})[1, ..., i-1]$, A'' outputs a random bit if the number of 1's in the string r is not equal to $n = n'/2$. Otherwise it outputs $A'(1^{nk}, G(s_{l_1})[1, ..., i-1] \oplus ... \oplus G(s_{l_n})[1, ..., i-1])$ where $l_1, ..., l_n$ are the indices such that $\forall l_j, r[l_j] = 1$.

Note that the string r has exactly n 1's with probability $\theta(1/\sqrt{n})$. This implies that the probability that A'' computes the above inner product is at least $(1/2 + \epsilon')$, where $\epsilon' = \theta(\epsilon/\sqrt{n}))$. Using averaging, we get that with probability at least $\epsilon'/2$ over the choice of $s_1, ..., s_{n'}$, A'' computes the inner product with a randomly chosen r with probability at least $1/2 + \epsilon'/2$. Now using the Goldreich-Levin Theorem (Theorem 2), we get that there is an algorithm A''' which for at least $\epsilon'/2$ fraction of $s_1, ..., s_{n'}$ computes $G(s_1), ..., G(s_{n'})$ with probability at least $\theta((\epsilon')^2)$. This implies that there is an algorithm which computes $G(s_1), ..., G(s_{n'})$ with probability $\Omega((\epsilon/\sqrt{n})^3)$ which gives us a contradiction from Lemma 14. □

Lemma 16 (unpredictability implies pseudorandomness). *Let $G : \{0,1\}^r \to \{0,1\}^{\ell(r)}$ be a generator such that for any probabilistic polynomial time algorithm A, the probability that A computes $G(s)[i]$ (for random s) given $G(s)[1...i-1]$ is at most $(1/2 + \delta)$. Then G is $(\delta \cdot \ell(r))$-pseudorandom.*

Proof. The proof follows from Theorem 3. □

Proof (proof of Theorem 6). Given the above Lemmas, we use the following sequence of arguments to prove the Theorem.

1. We use Yao's "pseudorandom implies unpredictable" reduction in Lemma 13 to argue that, for a random seed s, each output bit $G_{weak}(s)[i]$ (for $i \in [\ell(k)]$) is computable from the previous bits $G_{weak}(s)[1..(i-1)]$ with probability at most $1/2 + \delta$, which is some constant $\alpha < 1$ since $\delta < 1/2$ (this is where we need that $\delta < 1/2$).

2. We use the Direct-Product Lemma 14 to argue that, for each $i \in [\ell(k)]$, computing the direct-product $(G_{weak}(s_1)[i], ..., G_{weak}(s_{n'})[i])$ from $(G_{weak}(s_1)[1..(i-1)], ..., G_{weak}(s_{n'})[1..(i-1)])$ for *independent* random seeds $s_1, \ldots, s_{n'}$ can't be done better than with probability $\epsilon \leq e^{-\Omega(n')}$, which is negligible.
3. We use Lemma 15 to argue that, for each $i \in [\ell(k)]$, computing the XOR $G_{weak}(s_1)[i] \oplus \cdots \oplus G_{weak}(s_n)[i]$ (i.e., $G_{strong}(s_1, \ldots, s_n)[i]$) from the given bit-wise XOR of $G_{weak}(s_1)[1..(i-1)], ..., G_{weak}(s_n)[1..(i-1)]$ (i.e., from $G_{strong}(s_1, \ldots, s_n)[1..(i-1)]$), for independent random seeds $s_1, \ldots, s_n$, can't be done better than with probability $1/2+\mathrm{poly}(\epsilon n)$, which is negligibly better that random guessing.
4. Finally, we use Yao's "unpredictable implies pseudorandom" reduction in Lemma 16, to conclude that G_{strong} is $(\ell(k)\cdot\mathrm{poly}(\epsilon n))$-pseudorandom, which means that G_{strong} is $\delta'(k)$-pseudorandom for negligible $\delta'(k)$, as required.

□

4.2 Amplifying PRFs

Here we would like to show similar security amplification for pseudorandom function generators (PRFs).

First we recall the definition of a PRF. Let $\{f_s\}_{s\in\{0,1\}^*}$ be a function family, where, for each $s \in \{0,1\}^*$, we have $f_s : \{0,1\}^{d(|s|)} \to \{0,1\}^{r(|s|)}$. This function family is called *polynomial-time computable* if there is polynomial-time algorithm that on inputs s and $x \in \{0,1\}^{d(|s|)}$ computes $f_s(x)$. It is called $\delta(k)$-*pseudorandom function* family if, for every probabilistic polynomial-time oracle machine M, and all sufficiently large k, we have

$$|\mathbf{Pr}_s[M^{f_s}(1^k) = 1] - \mathbf{Pr}_{h_k}[M^{h_k}(1^k) = 1]| \leq \delta(k),$$

where s is chosen uniformly at random from $\{0,1\}^k$, and h_k is a uniformly random function from $\{0,1\}^{d(k)}$ to $\{0,1\}^{r(k)}$. Finally, we say that a PRF is *weak* if it is δ-pseudorandom for some constant $\delta < 1/2$, and we say a PRF is *strong* if it is $\delta(k)$-pseudorandom for some $\delta(k) < 1/k^c$ for any constant $c > 0$.

Let $\{f_s\}_s$ be a weak PRF. By analogy with the case of PRGs considered above, a natural idea for defining a strong PRF from $\{f_s\}_s$ is as follows: For some parameter n, take n independent seeds $\bar{s} = (s_1, \ldots, s_n)$, and define $g_{\bar{s}}(x)$ to be the bit-wise XOR of the strings $f_{s_1}(x), \ldots, f_{s_n}(x)$.

We will argue that the defined function family $\{g_{\bar{s}}\}_{\bar{s}}$ is a strong PRF. Rather than proving this directly, we find it more convenient to prove this first for the case of weak PRF $\{f_s\}_s$ of *Boolean* functions f_s, and use a simple reduction to get the result for general weak PRFs.

For the rest of this subsection, let $n > \omega(\log k)$ and let $n' = 2n$.

Theorem 7 (XOR Lemma for Boolean PRFs). *Let $\{f_s\}_s$ be a δ-pseudorandom Boolean function family for some constant $\delta < 1/2$. Let $\bar{s} = (s_1, \ldots, s_n)$ be a*

n-tuple of k-bit strings. Then, for some constant c_0 dependent on δ, the following function family $\{g_{\bar{s}}\}_{\bar{s}}$ is ϵ-pseudorandom for $\epsilon \leq \mathrm{poly}(k) \cdot e^{-(\delta n)/c_0}$:

$$g_{\bar{s}}(x) = f_{s_1}(x) \oplus \cdots \oplus f_{s_n}(x).$$

Proof. The idea is to view $\{f_s\}$ also as a MAC, which is a special case of a DWVP and hence we have a direct-product result (our Theorem 4). We will argue that if $g_{\bar{s}}$ is not a strong PRF, then one can break with non-negligible probability the direct product of MACs $(f_{s_1}, \ldots, f_{s_{n'}})$ for independent random seeds $s_1, \ldots, s_{n'}$, and hence (by Theorem 4), one can break a single MAC f_s with probability close to 1. The latter algorithm breaking f_s as a MAC will also be useful for breaking f_s as a PRF, with the distinguishing probability $\delta' > \delta$, which will contradict the assumed δ-pseudorandomness of the PRF $\{f_s\}_s$.

In more detail, suppose that A is a polynomial-time adversary that distinguishes $g_{\bar{s}}$ from a random function, with a distinguishing probability $\epsilon > \mathrm{poly}(k) \cdot e^{-\Omega(\delta n)}$. Using a standard hybrid argument, we may assume that the first query m of A is decisive. That is, answering this query with $g_{\bar{s}}(m)$ and all subsequent queries m_i with $g_{\bar{s}}(m_i)$ makes A accept with higher probability than answering this query randomly and all subsequent queries m_i with $g_{\bar{s}}(m_i)$. Let $\delta_1(k) \geq \epsilon/\mathrm{poly}(k)$ be the difference between the two probabilities.

Since $g_{\bar{s}}$ is a Boolean function, we can use Yao's "distinguisher-to-predictor" reduction [Yao82] to predict $g_{\bar{s}}(m)$ with probability $1/2 + \delta_1(n)$ over random n-tuples $\bar{s}$, and for the same fixed input m (since m is *independent* from the choice of $\bar{s}$).

By a standard argument, we get an algorithm A' for computing the following inner product

$$\langle f_{s_1}(m) \ldots f_{s_{n'}}(m), z \rangle, \tag{8}$$

for random $s_1, \ldots, s_{n'}$ and a random $z \in \{0,1\}^{n'}$, whose success probability is at least $1/2 + \delta_2(k) \geq 1/2 + \Omega(\delta_1(k)/\sqrt{n'})$; the idea is that a random $n' = 2n$-bit string z is balanced with probability $\Omega(1/\sqrt{n'})$, in which case we run the predictor for n-XOR, and otherwise (for non-balanced z) we flip a fair random coin. Next, by averaging, we get that, for at least $\delta_2(k)/2$ fraction of n'-tuples $s_1, \ldots, s_{n'}$, our algorithm A' correctly computes the inner product in (8) for at least $1/2 + \delta_2(k)/2$ fraction of random z's.

Applying the Goldreich-Levin algorithm from Theorem 2 to our algorithm A', we get an algorithm A'' that, for each of at least $\delta_2(k)/2$ fraction of n'-tuples $s_1, \ldots, s_{n'}$, computes $(f_{s_1}(m), \ldots, f_{s_{n'}}(m))$ with probability at least $\mathrm{poly}(\delta_2(k))$. Hence, this algorithm A'' computes $(f_{s_1}(m), \ldots, f_{s_{n'}}(m))$ for a non-negligible fraction of n'-tuples $s_1, \ldots, s_{n'}$.

Next, we view A'' as an algorithm breaking the n'-wise direct-product of the MAC f_s, with non-negligible probability. Using Theorem 4, we get from A'' an algorithm B that breaks the single instance of the MAC f_s with probability at least $1 - \delta'$ for $\delta' \leq O((\log(\mathrm{poly}(k)/\delta_2(k)))/n)$, which can be made less than $1/2 - \delta$ for $n > \omega(\log k)$ and sufficiently large constant c_0 in the bound on ϵ in the statement of the theorem (this is where we need the assumption that $\delta < 1/2$).

Note the algorithm B has $1 - \delta' > 1/2 + \delta$ probability over a secret key s to compute a correct message-tag pair (msg, tag) such that $f_s(msg) = tag$. Also note that the algorithm B makes some signing queries $f_s(q_i) =?$ for $q_i \neq msg$, but no verification queries (other than its final output pair (msg, tag)). We can use this algorithm B to distinguish $\{f_s\}_s$ from random in the obvious way: simulate B to get (msg, tag) (using the oracle function to answer the signing queries of B); query the oracle function on msg; if the answer is equal to tag, then accept, else reject.

Clearly, the described algorithm accepts with probability $1/2$ on a random oracle, and with probability greater than $1/2 + \delta$ on a pseudorandom function f_s. This contradicts the assumption that $\{f_s\}_s$ is δ-pseudorandom. □

As a corollary, we get the following.

Theorem 8 (Security amplification for PRFs). *Let $\{f_s\}_s$ be a weak PRF. For a parameter $n > \omega(\log k)$, take n independent seeds $\bar{s} = (s_1, \ldots, s_n)$, and define $g_{\bar{s}}(x)$ to be the bit-wise XOR of the strings $f_{s_1}(x), \ldots, f_{s_n}(x)$. The obtained function family $\{g_{\bar{s}}\}_{\bar{s}}$ is a strong PRF.*

Proof. Note that given a non-Boolean weak PRF $\{f_s\}_s$, we can define a Boolean function family $\{f'_s\}_s$ where $f'_s(x, i) = f_s(x)_i$, i.e., f'_s treats its input as an input x to f_s and an index $i \in [r(|s|)]$, and outputs the ith bit of $f_s(x)$. Clearly, if $\{f_s\}_s$ is δ-pseudorandom, then so is $\{f'_s\}_s$.

Then we amplify the security of $\{f'_s\}_s$, using our XOR Theorem for PRFs (Theorem 7). We obtain a strong PRF $\{g'_{\bar{s}}\}_{\bar{s}}$, where $\bar{s} = (s_1, \ldots, s_n)$ and $g'_{\bar{s}}(x, i) = f'_{s_1}(x, i) \oplus \cdots \oplus f'_{s_n}(x, i)$.

Finally, we observe that our function $g_{\bar{s}}(x)$ is the concatenation of the values $g'_{\bar{s}}(x, i)$ for all $1 \leq i \leq r(|k|)$. This function family $\{g_{\bar{s}}\}_{\bar{s}}$ is still a strong PRF, since we can simulate each oracle access to $g_{\bar{s}}$ with $d(|s|)$ oracle calls to $g'_{\bar{s}}$. □

5 Conclusions

We have established security amplification theorems for several interactive cryptographic primitives, including message authentication codes, digital signature and pseudorandom functions. The security amplifications for MACs and SIGs follow the direct product approach and work even for the weak variants of these primitives with imperfect completeness. For δ-pseudorandom PRFs, we have shown that the standard XOR lemma works for any $\delta < \frac{1}{2}$, which is optimal, complementing the non-standard XOR lemma of [Mye03], which works even for $\frac{1}{2} \leq \delta < 1$.

Of independent interest, we abstracted away the notion of dynamic weakly verifiable puzzles (DWVPs), which generalize a variety of known primitives, including ordinary WVPs, MACs and SIGs. We have also shown a very strong Chernoff-type security amplification theorem for DWVPs, and used it to establish our security amplification results for MACs, SIGs and PRFs.

Acknowledgments: Yevgeniy Dodis was supported in part by NSF Grants 0831299, 0716690, 0515121, 0133806. Part of this work was done while the author was visiting the Center for Research on Computation and Society at Harvard University. Russell Impagliazzo was supported in part NSF Grants 0716790, 0835373, 0832797, and by the Ellentuck Foundation. Ragesh Jaiswal was supported in part by NSF Grant 0716790, and completed part of this work while being at the University of California at San Diego.

References

[BIN97] M. Bellare, R. Impagliazzo, and M. Naor. Does parallel repetition lower the error in computationally sound protocols? In *Proceedings of the Thirty-Eighth Annual IEEE Symposium on Foundations of Computer Science*, pages 374–383, 1997.

[CHS05] R. Canetti, S. Halevi, and M. Steiner. Hardness amplification of weakly verifiable puzzles. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005*, pages 17–33, 2005.

[Cor00] J.S. Coron. On the exact security of full domain hash. In *Advances in Cryptology - CRYPTO 2000, Twentieth Annual International Cryptology Conference*, pages 229–235, 2000.

[CRS+07] R. Canetti, R. Rivest, M. Sudan, L. Trevisan, S. Vadhan, and H. Wee. Amplifying collision resistance: A complexity-theoretic treatment. In *Advances in Cryptology - CRYPTO 2007, Twenty-Seventh Annual International Cryptology Conference*, pages 264–283, 2007.

[DNR04] C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques*, pages 342–360, 2004.

[GL89] O. Goldreich and L.A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-Lemma. *Electronic Colloquium on Computational Complexity*, TR95-050, 1995.

[Gol01] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, 2001.

[IJK08] R. Impagliazzo, R. Jaiswal, and V. Kabanets. Chernoff-type direct product theorems. *Journal of Cryptology*, 2008. (published online September 2008); preliminary version in CRYPTO'07.

[IJKW08] R. Impagliazzo, R. Jaiswal, V. Kabanets, and A. 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.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the Thirty-Sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[Lev87] L.A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[LR86] M. Luby and C. Rackoff. Pseudorandom permutation generators and cryptographic composition. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 356–363, 1986.

[Mye03] S. Myers. Efficient Amplification of the Security of Weak Pseudo-Random Function Generators. In *J. Cryptology*, 16(1):1–24, 2003.

[Mye99] S. Myers. On the development of block-ciphers and pseudorandom function generators using the composition and XOR operators. Master's thesis, University of Toronto, 1999.

[NR98] M. Naor and O. Reingold. From unpredictability to indistinguishability: A simple construction of pseudo-random functions from MACs. In *Advances in Cryptology - CRYPTO 1998, Eigtheenth Annual International Cryptology Conference*, pages 267–282, 1998.

[NR99] M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, pages 29–66, 1999.

[PV07] R. Pass and M. Venkitasubramaniam. An efficient parallel repetition theorem for Arthur-Merlin games. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 420–429, 2007.

[PW07] K. Pietrzak and D. Wikstrom. Parallel repetition of computationally sound protocols revisited. In *Theory of Cryptography, Fourth Theory of Cryptography Conference, TCC 2007*, pages 86–102, 2007.

[Yao82] A.C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.