Balancing Output Length and Query Bound in Hardness Preserving Constructions of Pseudorandom Functions[*]

Nishanth Chandran[†] Sanjam Garg[‡]

Abstract

We revisit hardness-preserving constructions of a pseudo-random function (PRF) from any length doubling pseudo-random generator (PRG) when there is a non-trivial upper bound q on the number of queries that the adversary can make to the PRF. Very recently, Jain, Pietrzak, and Tentes (TCC 2012) gave a hardness-preserving construction of a PRF that makes only $O(\log q)$ calls to the underlying PRG when $q = 2^{n^\epsilon}$ and $\epsilon \geq \frac{1}{2}$. This dramatically improves upon the efficiency of the construction of Goldreich, Goldwasser, and Micali (FOCS 1984). However, they explicitly left open the question of whether such constructions exist when $\epsilon < \frac{1}{2}$. In this work, we give constructions of PRFs that make only $O(\log q)$ calls to the underlying PRG when $q = 2^{n^\epsilon}$, for $0 < \epsilon < 1$; our PRF outputs $O(n^{2\epsilon})$ bits (on every input), as opposed to the construction of Jain *et al.* that outputs n bits. That is, our PRF is not length preserving; however it outputs more bits than the PRF of Jain *et al.* when $\epsilon > \frac{1}{2}$. We obtain our construction through the use of information-theoretic tools such as *almost* α-wise independent hash functions coupled with a novel proof strategy.

1 Introduction

Pseudo-random functions. Goldreich, Goldwasser, and Micali introduced the fundamental notion of pseudo-random functions in their seminal paper [12]. A pseudo-random function (PRF) is a keyed function $\mathsf{F} : \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^m$ (with key length ℓ and input length u) such that no efficient adversary, that can make adaptive oracle queries to F, can distinguish between the outputs of F and a uniformly random function (from u bits to n bits). Pseudo-random functions have found extensive applications in cryptography - from symmetric-key cryptography to software protection to zero-knowledge proofs [11, 10, 9, 17, 16].

Goldreich *et al.* showed how to construct a PRF from any length doubling *pseudo-random generator*, that is a function $\mathsf{G} : \{0,1\}^n \to \{0,1\}^{2n}$ that takes as input a seed of n bits and outputs $2n$ bits that are computationally indistinguishable from the uniform distribution on $2n$ bits. We will refer to their construction as the GGM construction. The GGM construction gives rise to a PRF $\mathsf{GGM}_\mathsf{G} : \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^m$ for $\ell = n$ and $m = n$ (in fact this construction works for any input length $u \in \mathbb{N}$ with other parameters remaining n).

Hardness preservation. Now, assume that the underlying PRG G has "σ bits of security", meaning that no adversary of size 2^σ can distinguish $\mathsf{G}(U_n)$ from U_{2n} (where U_t denotes the uniform distribution on t bits) with advantage more than $2^{-\sigma}$. The GGM construction of a PRF is *hardness preserving*, i.e., if G has cn bits of security (for some constant $c > 0$ and all sufficiently large n), then GGM_G has $c'n$ bits of security for some $0 < c' \leq c$. The domain size of the PRF can be arbitrary ($\{0,1\}^u$), but the GGM construction makes u calls to the underlying PRG, and hence the efficiency of GGM_G depends critically on u. Levin [15] suggested a trick that improves the efficiency when u is large; first hash the u bits down to v bits, using a universal hash function, and now apply the GGM construction to the v

[*] Full version of paper published at Indocrypt 2014

[†] Microsoft Research, India; Email: `nichandr@microsoft.com`. Work done while this author was at Microsoft Research, Redmond.

[‡] University of California, Berkeley; Email: `sanjamg@berkeley.edu`. Work done while this author was visiting Microsoft Research, Redmond. Part of this research conducted while at the IBM Research, T.J. Watson funded by NSF Grant No.1017660.

bits obtained. Now, the efficiency of the GGM construction can be reduced to v calls to G, and v can be set to $\omega(\log n)$, if we want security only against polynomial-size adversaries. However, if we care about preserving hardness (if G has cn bits of security), then we are forced to set $v = \Omega(n)$ and hence, the best hardness-preserving construction of a PRF, F, from a length-doubling PRG, G, requires $\Theta(n)$ calls to G (except in the case when $u = o(n)$ is sublinear, in which case one can use the GGM construction directly).

Constructing PRFs with bound on queries of adversary. Jain, Pietrzak, and Tentes [13] considered a setting, in which the size of the adversary is still 2^{σ} (or 2^{cn} when G is exponentially hard), but there is a bound q on the number of queries that the adversary can make to the PRF F. Surprisingly, by making such a restriction on the adversary, they can dramatically improve the efficiency of PRF constructions. In more detail, they consider an adversary of size 2^{cn} who can make only $q = 2^{n^{\epsilon}}$ (for constant $\frac{1}{2} \leq \epsilon < 1$) queries to F; against such an adversary, they give a construction of a PRF F that only makes $\Theta(n^{\epsilon})$ calls to the underlying PRG, but is still hardness preserving.

Hardness preservation and bounding queries in practice. We stress here that the notion of hardness preserving constructions of PRFs is important both in theory as well as in practice. For example, if we have a hardness preserving construction of a PRF (and the underlying PRG has exponential security) we can scale down the security parameter by almost a logarithmic factor in practical scenarios where security against only polynomial-size adversaries is required. This implies an almost exponential improvement in the efficiency. Furthermore, in most practical situations, we will be able to bound the number of queries that the adversary can make to the PRF; e.g., if we are using the PRF for constructing a symmetric encryption scheme, it is conceivable that we can bound the number of ciphertexts that the adversary can get to see (which inherently bounds the number of queries to the PRF that the adversary can get). In such situations, we can obtain much more efficient hardness preserving constructions of PRFs.

1.1 Our Results

Unfortunately, for the case when $\epsilon < \frac{1}{2}$, that is when $q = 2^{o(\sqrt{n})}$, the construction of Jain *et al.* [13] does not offer any further improvement in efficiency, and the best hardness preserving construction of F from G makes $\Theta(\sqrt{n})$ calls to G.

In this work, we are precisely interested in answering the following question: can we obtain a hardness preserving construction of a PRF from a PRG that makes $o(\sqrt{n})$ calls to G, when q, the bound on the number of queries that the adversary can make to the PRF, is $2^{o(\sqrt{n})}$?

Roughly speaking it seems that, as any hardness preserving construction of a PRF tries to make fewer calls to the underlying PRG, its ability to output pseudo-random bits reduces.[1] In the setting where a PRF from a PRG makes $o(\sqrt{n})$ calls to G and when q is bounded by $2^{o(\sqrt{n})}$ we present a tradeoff between the output length of the PRF, the bound q, and the level of hardness preserved. In particular:

- We provide a hardness preserving construction of a PRF $\mathsf{F} : \{0,1\}^{\ell} \times \{0,1\}^{n} \to \{0,1\}^{n^{2\epsilon}}$ (where $\ell = \Theta(n)$ denotes the seed length, n denotes the input length, and $n^{2\epsilon}$ denotes the output length), that makes $\Theta(n^{\epsilon})$ calls to $\mathsf{G} : \{0,1\}^{n} \to \{0,1\}^{2n}$ for any $u \in \mathbb{N}$, where $q = 2^{n^{\epsilon}}$ and $0 < \epsilon < 1$. Note that the PRF in this construction outputs $n^{2\epsilon}$ bits as opposed to n bits as in the construction of Jain *et al.* [13]. For the case when $\epsilon > \frac{1}{2}$, our construction outputs more number of random bits than the construction of Jain *et al.* [13], while making the same number of calls to G.

- Next, we note that if we wished to output more number of bits (say n, as opposed to $n^{2\epsilon}$, when $\epsilon < \frac{1}{2}$), then we can obtain a construction that makes the same $\Theta(n^{\epsilon})$ calls to G, but the resulting PRF $\mathsf{F} : \{0,1\}^{\ell} \times \{0,1\}^{n} \to \{0,1\}^{n}$ has only $n^{2\epsilon}$ bits of security.
  Alternatively, we can obtain a construction of a PRF with cn bits of hardness and n bits of output by repeating the construction of $\mathsf{F} : \{0,1\}^{\ell} \times \{0,1\}^{n} \to \{0,1\}^{n^{2\epsilon}}$, $n^{1-2\epsilon}$ times in parallel. While the total number of calls to the PRG made by this construction will be $\Theta(n^{1-\epsilon})$ (which is larger than $\Theta(n^{\epsilon})$ for $0 < \epsilon < \frac{1}{2}$), the advantage of this construction is that the depth of the circuit evaluating our PRF will still be $\Theta(n^{\epsilon})$.

[1] In fact with appropriate formalization Jain *et al.* conjecture this.

- We also note that, just as in [13], our results extend to the case when we start with a PRG G that has only $\sigma = o(n)$ bits of hardness, and we have a bound $q = 2^{\sigma^{\epsilon}}$ on the number of queries that the adversary can make to the PRF, when $0 < \epsilon < 1$.[2]
- Finally, we mention that, similar to Jain *et al.* [13], our techniques can be used to give more efficient constructions in other settings; for example by applying it to the work of Naor and Reingold [21] who construct PRFs computable in low depth from pseudorandom synthesizers (objects stronger than PRGs but weaker than PRFs). The construction of [21] gives a hardness-preserving construction of a PRF from a pseudorandom synthesizer (PRS) with $\Theta(n)$ calls to the PRS in depth $\Theta(\log n)$. Our techniques can be used to improve the efficiency of their construction to $\Theta(\log q)$ whenever one can put an upper bound $q = 2^{n^{\epsilon}}, 0 < \epsilon < 1$ on the number of adversarial queries.

Concurrent and independent work. Concurrently and independently of our work, Berman *et al.* [4], using very different techniques based on cuckoo hashing, show how to construct a hardness preserving PRF from n bits to n bits that makes $\mathcal{O}(n^{\epsilon})$ calls to the underlying PRG when the bound on the number of queries made by the adversary is $2^{n^{\epsilon}}$, for any $0 < \epsilon < 1$. A technical comparison between our work and the work of Berman *et al.* [4] follows:

- In our construction the seed length of the PRF contructed, ℓ, is $\Theta(n)$, while the PRF construction of Berman *et al.* [4] requires a seed of length $\Theta(n^2)$.
- Secondly, in the case when $\epsilon < \frac{1}{2}$, their PRF outputs more random bits (n) than our construction ($n^{2\epsilon}$); however, when $\epsilon > \frac{1}{2}$, the situation is reversed, and our PRF outputs more random bits ($n^{2\epsilon}$) as opposed to their construction (as well as [13], since both constructions output n bits) while making the same number of GGM calls.

1.2 Technical Difficulties and New Ideas

In order to present the high level ideas behind our construction, we shall begin by describing Levin's trick and how Jain *et al.* build upon it to construct a PRF that makes fewer calls to the PRG.

Background. Recall that the key idea behind Levin's trick [15] is to first hash the input bits down using a (information-theoretically secure) universal hash function $h : \{0,1\}^u \to \{0,1\}^v$, and then applying the GGM construction to the obtained hashed bits. The efficiency of the GGM construction depends on the output length of this hash function. Now observe that that in order to use Levin's trick, to obtain a hardness preserving construction, one must set $v = \Theta(n)$. If v were to be $o(n)$, then the probability that two inputs x_i and x_j collide on the hash function's output would be $2^{-o(n)}$. In other words $\mathsf{F}(x_i) = \mathsf{F}(x_j)$ (since $\mathsf{F}(x) = \mathsf{GGM}_G(k, h(x))$) and this would prevent us from achieving exponential security. Hence we need that a hardness preserving construction of a PRF makes $\Theta(n)$ calls to the underlying PRG.

The key idea of Jain *et al.* is to exploit tools from information-theoretic t-wise independent hash functions and hash the bits using a hash function h_1 to $2t$ bits anyway (t is chosen to be $\log q$, where $q = 2^{\sqrt{n}}$ is the bound on the number of queries that the adversary can make to F) and then evaluate the GGM construction on it.[3] Of course there will be collisions and so they do not output the generated value directly. Instead they use the output in order to derive a key for another hash function h_2 which is then applied on the original un-hashed input. The resulting value is the output. The derivation of the key from the output of the GGM function involves stretching the output of the GGM function using the PRG. In particular, the output of GGM is stretched by a factor of t so that it is large enough to serve as the seed for h_2.

Roughly speaking if h_1 and h_2 are both t-wise independent hash functions, then Jain *et al.* can argue the security of their scheme as long as the adversary makes less than q queries to the PRF. The crucial idea is that since h_1 is t-wise independent not too many inputs (specifically no more than t) can collide on the first hashing. Furthermore the few (up to t) that do happen to collide on a specific value will ultimately lead to random outputs as h_2 is also t-wise independent. More formally, h_1 is a t-wise independent hash functions,

[2] For clarity of exposition, we present our results only in the case when G has exponential hardness.

[3] We consider only the case where $q = 2^{\sqrt{n}}$ in the discussion below, but the argument holds for $q = 2^{n^{\epsilon}}$, for $\frac{1}{2} \leq \epsilon < 1$.

and therefore the probability that one gets a $t+1$-wise collision after the first hashing (i.e., some $t+1$ of the q queries hash down to the same value) is upper bounded roughly by $\frac{1}{2^{t^2}}$, which is exponentially small, when $q \geq 2^{\sqrt{n}}$. This implies that with very high probability, at most t inputs will use the same seed for the t-wise independent hash function h_2 and hence the outputs will be pseudorandom.

Now, observe that the total number of calls that the above PRF construction makes to the underlying PRG is $\Theta(t)$. This is because the GGM construction is executed on a t-bit input (generated as the output of the hash function h_1) which makes t calls to the underlying PRG. The construction additionally makes t calls in deriving the key for h_2 from the output of GGM.

Overview of our construction: new ideas. In order to reduce the number of calls in the above construction, we need to reduce the number of calls that F makes in two places: the GGM part, as well as the PRG stretch (to sample the t-wise independent hash function h_2) part. We are interested in the case when $q = 2^{n^\epsilon}$, for $0 < \epsilon < \frac{1}{2}$ and would like to obtain a construction that makes $\Theta(n^\epsilon)$ calls to the PRG. However in order to make just $\Theta(n^\epsilon)$ calls to the PRG we need to reduce the output length of the first hash function h_1 to $\Theta(n^\epsilon)$. Recall that the probability of a $t+1$-wise collision is bounded by roughly $\frac{1}{2^{t^2}}$, which is sub-exponential (when $\epsilon < \frac{1}{2}$) for our setting. Another problem is that far too many inputs to the F will collide in the key of h_2 than what h_2 is equipped to handle. The only option seems to be to make the hash functions much more resilient to collisions, or in other words increase the parameter t for both h_1 and h_2. That is, we use α-wise independent hash functions h_1 and h_2 for some parameter $\alpha = \omega(t)$. However this is fundamentally problematic since the GGM part of the construction outputs n bits that need to be "stretched" to get a seed of length $\Theta(\alpha n)$ bits. This, unfortunately, would end up requiring $\omega(t)$ calls to G for stretching the output of GGM from n bits to $\Theta(\alpha n)$.

Our key idea here is to use approximate constructions of α-wise independent hash functions [20, 1, 14] for h_1 and h_2. The key advantage of these hash functions is that they can be constructed using roughly $\Theta(m\alpha)$ bits of randomness (where m is the *output* length), instead of the $\Theta(n\alpha)$ bits needed for perfect constructions (where n is the *input* length). Hence by decreasing the output length we can obtain the desired level of resilience to collisions. This allows us to obtain a tradeoff between the efficiency of the PRF and the length of the output it generates.

Even though our construction makes a seemingly simple tweak to the construction of Jain *et al.* [13], unfortunately their proof strategy does not work for us. The fundamental reason behind this is that having a perfect t-wise independent hash function allows them to reduce an adaptive distinguisher directly to a non-adaptive distinguisher. Intuitively speaking, this follows from the fact that the outputs of the t-wise independent hash function are uniformly random strings and hence when these values are used as the outputs of the PRF, they prove useless for the adaptive distinguisher. Formally, this follows from a claim of Maurer [18]. However, in our setting, the responses are not uniform and the slight bias could help the adversary choose its queries intelligently, triggering the events that ultimately allow it to distinguish the function from random. This prevents us from using the results of Maurer [18]. It is worth noting here, that the problem of constructing adaptively secure pseudorandom functions from non-adaptive pseudorandom functions [19, 22, 23, 8, 2, 4] is a very important problem that has received plenty of attention.

For our construction, we prove security against adaptive distinguishers using a *step-ladder* approach. More specifically, consider an adaptive distinguisher that makes i queries to the PRF. Its distinguishing advantage can be used to upper bound the statistical distance between the distribution of the responses to i adaptive queries and the distribution of i uniform strings. This statistical difference gives us an upper bound on the advantage an adaptive distinguisher has in the choice it makes for the $i+1^{th}$ query over its non-adaptive counterpart. Given this we can evaluate the distinguishing advantage of an adaptive distinguisher that makes $i+1$ adaptive queries. Carefully applying this process repeatedly allows us to obtain a bound on the distinguishing advantage of an adaptive distinguisher making q queries.

Finally, we remark that our various results are obtained by setting α and m (output length of F) appropriately, according to the hardness of G and q.

1.3 Roadmap

We start by recalling the preliminary notions and definitions needed in Section 2. Then we provide our construction in Section 3 and the proof of our main theorem in Section 4. Finally we conclude in Section 5.

2 Preliminaries

In this section we recall and define some basic notions and setup notation. Let λ denote the security parameter. We say that a function is *negligible* in the security parameter if it is asymptotically smaller than the inverse of any fixed polynomial. Otherwise, the function is said to be *non-negligible* in λ. We say that an event happens with *overwhelming* probability if it happens with a probability $p(\lambda) = 1 - \nu(\lambda)$ where $\nu(\lambda)$ is a negligible function in λ.

Notation. We denote values and bit strings by lower case letters and random variables by uppercase letters. Sets are denoted by uppercase calligraphic letters. We use U_n to denote the random variable which takes values uniformly at random from the set of n bit long strings and $\mathcal{R}_{n,m}$ to denote the set of all functions $F : \{0,1\}^n \mapsto \{0,1\}^m$. For a set $\mathcal{X}$, $\mathcal{X}^t$ denotes the t'th direct product of $\mathcal{X}$, i.e., $(\mathcal{X}_1, \dots, \mathcal{X}_t)$ of t identical copies of $\mathcal{X}$ and for a random variable X, $X^{(t)}$ denotes the random variable which consists of t independent copies of X. Let $x \leftarrow X$ denote the fact that x was chosen according to the random variable X and analogously let $x \leftarrow \mathcal{X}$ denote that x can chosen uniformly at random from set $\mathcal{X}$. For random variables X_0, X_1 distributed over a set $\mathcal{X}$, we use $X_0 \sim X_1$ to denote that the random variables are identically distributed, we use $X_0 \sim_\delta X_1$ to denote that they have a statistical distance δ, i.e. $\frac{1}{2}\sum_{x\in\mathcal{X}} \mid \mathsf{Pr}_{X_0}[x] - \mathsf{Pr}_{X_1}[x] \mid \leq \delta$, and finally we use $X_0 \sim_{(\delta,s)} X_1$ to denote that they are (δ, s) indistinguishable, i.e. for all distinguishers D of size at most $|D| \leq s$ we have $\mid \mathsf{Pr}_{X_0}[D(x) \rightarrow 1] - \mathsf{Pr}_{X_1}[D(x) \rightarrow 1] \mid \leq \delta$.

2.1 Pseudorandom Functions

We recall the definitions of pseudorandom generators (PRG) and pseudorandom functions (PRF). Subsequently we will describe the GGM construction of a pseudorandom function from a pseudorandom generator.

Definition 1 (PRG [5, 25]) *A length-increasing function* $\mathsf{G} : \{0,1\}^n \mapsto \{0,1\}^m$ *where* $m > n$ *is a* (δ, s)*-hard pseudorandom generator if*

$$\mathsf{G}(U_n) \sim_{(\delta,s)} U_m$$

We say that G *has* σ *bits of security if* G *is* $(2^{-\sigma}, 2^\sigma)$*-hard.* G *is exponentially hard if it has* cn *bits of security for some* $c > 0$*, and* G *is sub-exponentially hard if it has* cn^ϵ *bits of security for some* $c > 0, 0 < \epsilon < 1$.

Stretching a PRG. Let $\mathsf{G} : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ be a length doubling function. For $e \in \mathbb{N}$, let $\mathsf{G}^e : \{0,1\}^n \rightarrow \{0,1\}^{en}$ be the function that takes an n bit string as input and expands it to an en bit string using $e-1$ invocations of G. This can be done sequentially or via a more efficient parallel computation of depth $\lceil \log e \rceil$. We now have the following lemma.

Lemma 1 *As stated in* **[13]**. *Let* G *be a* (δ, s)*-hard PRG, then* G^e *is a* $(e \cdot \delta, s - e \cdot |\mathsf{G}|)$*-hard PRG.*

Definition 2 (PRF [12]) [4] *A function* $\mathsf{F} : \{0,1\}^\ell \times \{0,1\}^n \rightarrow \{0,1\}^m$ *is a* (q, δ, s)*-hard pseudorandom function (PRF) if for every oracle aided distinguisher* D^* *of size* $|D^*| \leq s$ *making at most* q *oracle queries*

$$\mid \mathsf{Pr}_{k\leftarrow\{0,1\}^\ell}[D^{\mathsf{F}(k,\cdot)} \rightarrow 1] - \mathsf{Pr}_{f\leftarrow\mathcal{R}_{n,m}}[D^{f(\cdot)} \rightarrow 1] \mid \leq \delta$$

F *has* σ *bits of security against* q *queries if* F *is* $(q, 2^{-\sigma}, 2^\sigma)$ *secure. If* q *is unspecified then it is assumed to be unbounded (the size* 2^σ *of the distinguisher is a trivial upper bound on* q*).*

[4] We use the specific definition of [13].

The GGM Construction. Goldreich, Goldwasser and Micali [12] gave the first construction of a pseudorandom function from any length doubling PRG. Their construction is described below. For any length doubling PRG $\mathsf{G} : \{0,1\}^n \to \{0,1\}^{2n}$, $\mathsf{GGM}_\mathsf{G} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is defined as a function that takes as input x along with a seed k. The output of the function $\mathsf{GGM}_\mathsf{G}(k, x)$ is k_x that can be obtained by recursive evaluation using $k_\epsilon = k$ and $k_{a||0}||k_{a||1} := \mathsf{G}(k_a)$ (where ϵ denotes the empty string).

Proposition 1 (PRF [12]) *If G is a (δ_G, s_G)-hard PRG, then for any $n, q \in \mathbb{N}$, $\mathsf{GGM}_\mathsf{G} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a (q, δ, s)-hard PRF where*

$$\delta = n \cdot q \cdot \delta_\mathsf{G} \quad s = s_\mathsf{G} - q \cdot n \cdot |\mathsf{G}|$$

We remark, that in general, using the above same transformation, one can also obtain a PRF $\mathsf{GGM}_\mathsf{G} : \{0,1\}^n \times \{0,1\}^u \to \{0,1\}^n$ for any $u \in \mathbb{N}$. We will use GGM_G to refer to the PRF so obtained (with input length u) when the value of u is clear from context.

2.2 Information theoretic tools

The construction of pseudorandom functions presented in this paper relies on some well studied information theoretic tools. Next we recall these notions.

Definition 3 (α-wise independence [6, 24]) *For $\ell, m, n, \alpha \in \mathbb{Z}$, a function $h : \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^m$ is α-wise independent, if for every α distinct inputs $x_1, \ldots, x_\alpha \in \{0,1\}^n$ and a random key $k \leftarrow \{0,1\}^\ell$ the outputs are uniform, i.e.*

$$h_k(x_1)|| \ldots ||h_k(x_\alpha) \sim U_m^{(\alpha)}$$

Proposition 2 *For any $\alpha, n, m \leq n$ there exists an α-wise independent hash function with key length $\ell = n \cdot \alpha$.*

Definition 4 (Almost α-wise independence [20]) *For $\ell, m, n, \alpha \in \mathbb{Z}$, a function $h : \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^m$ is (δ, α)-wise independent, if for every α distinct inputs $x_1, \ldots, x_\alpha \in \{0,1\}^n$ and a random key $k \leftarrow \{0,1\}^\ell$ the outputs are statistically close to uniform, i.e.*

$$h_k(x_1)|| \ldots ||h_k(x_\alpha) \sim_\delta U_m^{(\alpha)}$$

Proposition 3 ([1, 14]) *For any α, n, m there exists a (δ, α)-wise independent hash function with key length $\ell = O(m\alpha + \log \frac{n}{\delta})$.*[5]

3 Our Construction

Our construction will use two parameters q, α. Recall that q represents the bound on the number of queries to the PRF that the adversary is allowed to make. We will use t as a shorthand for the value $\log q$. On the other hand α is a parameter that will depend on the other parameters in the system. Very roughly looking ahead α will need to increase as the desired level of security increases.

We use a $(\delta_1, \alpha + 1)$-wise independent hash function $h_1 : \{0,1\}^{\ell_1} \times \{0,1\}^n \to \{0,1\}^{2t}$ with appropriate seed length $\ell_1 = O(t\alpha + \log \frac{n}{\delta_1})$. (cf. Proposition 3) We will also need a $(\delta_2, \alpha + 1)$-wise independent hash function $h_2 : \{0,1\}^{\ell_2} \times \{0,1\}^n \to \{0,1\}^m$ with appropriate seed length $\ell_2 = O(m\alpha + \log \frac{n}{\delta_2})$.

Let $\mathsf{C}^\mathsf{G} : \{0,1\}^{\ell_1 + n} \times \{0,1\}^n \to \{0,1\}^m$ be our PRF that on input a key $k = k_0||k_1$(where $k_0 \in \{0,1\}^{\ell_1}$ and $k_1 \in \{0,1\}^n$) and $x \in \{0,1\}^n$, computes the output as:

$$\mathsf{C}(k, x) = h_2(\mathsf{G}^{2t}(\mathsf{GGM}_\mathsf{G}(k_1, h_1(k_0, x))), x).$$

[5] To see that this is true, use Theorem 3 from [1] and consider the construction that outputs $2^n \cdot m$ bits which are δ-away (in L_1 norm) from $m\alpha$-wise independence using roughly $m\alpha + 2\log(\frac{\alpha \log(2^n \cdot m)}{2\delta})$ bits. This gives us the desired hash function.

Input	Query bound	Output	Hardness δ
n	q	$(\log q)^2$	2^{-cn}
n	q	$\sqrt{n}\log q$	$2^{-\sqrt{n}\log q}$
n	$q < 2^{\sqrt{n}}$	n	$2^{-(\log q)^2}$
n	$q \geq 2^{\sqrt{n}}$	$(\log q)^2$	2^{-cn}

Figure 1: Security obtained for different settings of parameters.

Theorem 1 (Main Theorem) *If G is a $(\delta_{\mathsf{G}}, s_{\mathsf{G}})$-hard PRG, then C^{G} is a (q, δ, s)-secure PRF where*

$$\delta \leq 4 \cdot q \cdot t \cdot \delta_{\mathsf{G}} + \frac{q^2}{2^n} + \frac{q^2}{2^{t\alpha}} + q^2 \cdot 2^{t\alpha}(\delta_1 + q \cdot \delta_2) + q \cdot \delta_2$$

$$m\alpha + \log \frac{n}{\delta_2} \leq ctn$$

where $c > 0$ is an appropriately chosen constant and $t = \log q$. Finally note that the seed length, $\ell = O(t\alpha + \log \frac{n}{\delta_1} + n)$ and a total of $\Theta(t)$ calls are made to G.

$$s = s_{\mathsf{G}} - q \cdot |C^{\mathsf{G}}| - 2q \cdot t \cdot |\mathsf{G}|$$

Implications of Theorem 1. Now, suppose we want to obtain a hardness-preserving construction to obtain a PRF with $c'n$ bits of hardness, then in order to obtain $\delta = 2^{-c'n}$, we must set $t\alpha = n$. This means that $m \approx t^2$ (from the constraint $m\alpha + \log \frac{n}{\delta_2} \leq ctn$); in other words we obtain a hardness-preserving construction of a PRF that outputs $n^{2\epsilon}$ bits. At the other end of the spectrum, suppose we want the PRF to output n bits, then we must set $\alpha \approx t$, which gives us a PRF with $n^{2\epsilon}$ bits of hardness. To give a better perspective of the various results obtained by setting the values of α and m appropriately, we give some examples for the choices of these parameters in Figure 1.

4 Proof of Theorem 1

We prove Theorem 1 by considering the following sequence of hybrids $\mathcal{H}_0, \mathcal{H}_1 \ldots, \mathcal{H}_4$. In hybrid $\mathcal{H}_i$ (for $i \in \{0, 1, \ldots 4\}$) samples are generated according to the circuit C_i (described in the sequel). In the first hybrid, C_0 corresponds to the execution of C itself and in the last hybrid $\mathcal{H}_4$, C_4 corresponds to the random function $\mathcal{R}_{n,m}$.

[-] **Hybrid $\mathcal{H}_0$:** This hybrid corresponds to actual evaluation of the function C. In other words $\mathsf{C}_0(k, x) = \mathsf{C}(k, x) = h_2(\mathsf{G}^{2t}(\mathsf{GGM}_{\mathsf{G}}(k_1, h_1(k_0, x))), x)$. For any machine D^{C_0} of size s and that makes q queries to C_0 let $p_0^D = \mathsf{Pr}[D^{\mathsf{C}_0(\cdot)} = 1]$.

[-] **Hybrid $\mathcal{H}_1$:** This hybrid is the same as the previous hybrid except that we use a random function $\mathcal{R}_{2t,n}$ instead of the $\mathsf{GGM}_{\mathsf{G}}$ execution. More specifically, $\mathsf{C}_1(k, x) = h_2(\mathsf{G}^{2t}(\mathcal{R}_{2t,n}(h_1(k_0, x))), x)$. For any machine D^{C_1} of size s and that makes q queries to C_1 let $p_1^D = \mathsf{Pr}[D^{\mathsf{C}_1(\cdot)} = 1]$. We now show:

Lemma 2 *For every adversarial q-query distinguisher D of size $s \leq s_{\mathsf{G}} - 2q \cdot t \cdot |\mathsf{G}| - q \cdot |C^{\mathsf{G}}|$ we have that $|p_1^D - p_0^D| \leq 2q \cdot t \cdot \delta_{\mathsf{G}}$.*

Proof. Assume $|p_0^D - p_1^D| > 2q \cdot t \cdot \delta_{\mathsf{G}}$. Then we construct a distinguisher D' of size at most $s + q \cdot |C^{\mathsf{G}}|$ (which is equal to $s_{\mathsf{G}} - 2q \cdot t \cdot |\mathsf{G}|$), that distinguishes $\mathsf{GGM}_{\mathsf{G}}$ from $\mathcal{R}_{2t,n}$ with a distinguishing advantage $> 2q \cdot t \cdot \delta_{\mathsf{G}}$, leading to a contradiction to Proposition 1. D' has access to an oracle $\mathcal{O}(\cdot)$ that generates a sample according to $\mathsf{GGM}_{\mathsf{G}}(k_1, \cdot)$ (for a random k_1) or according to $\mathcal{R}_{2t,n}$ and it needs to distinguish among the two.

Now we describe our distinguisher D'. D' samples a random seed k_0 of appropriate length. It executes D internally that makes queries for C. Consider a query x. Let y be the response of $\mathcal{O}$ on the query $h_1(k_0, x)$. D' responds with the value $h_2(\mathsf{G}^{2t}(y, x))$ to D. Observe that the responses of D' to D correspond to evaluations of the circuit C_0 if the oracle $\mathcal{O}(\cdot)$ samples according to the distribution $\mathsf{GGM}_\mathsf{G}(k_1, \cdot)$ (for a random k_1) and to evaluations of the circuit C_1 if the oracle samples according to $\mathcal{R}_{2t,n}$. Hence the success of D is distinguishing between the two cases directly translates to the success of D' in distinguishing GGM_G from $\mathcal{R}_{2t,n}$. Note that D' makes q queries to $\mathcal{O}$ which is same as the number of queries D makes. Note that the size of our distinguisher D' is larger than the size of D by at most $q \cdot |C^\mathsf{G}|$. ■

[-] **Hybrid $\mathcal{H}_2$:** This hybrid is the same as the previous hybrid except that we use a random function $\mathcal{R}_{n,2tn}$ instead of executing G^{2t}. More specifically, $\mathsf{C}_2(k, x) = h_2(\mathcal{R}_{n,2tn}(\mathcal{R}_{2t,n}(h_1(k_0, x))), x)$. For any machine D^{C_2} of size s and that makes q queries to C_2, let $p_2^D = \Pr[D^{\mathsf{C}_2(\cdot)} = 1]$. Now, we show:

Lemma 3 *For any adversarial q-query distinguisher D of size $s = s_\mathsf{G} - 2q \cdot t \cdot |\mathsf{G}| - q \cdot |C^\mathsf{G}|$ we have that $|p_2^D - p_1^D| \leq 2q \cdot t \cdot \delta_\mathsf{G}$.*

Proof. Assume $|p_0^D - p_1^D| > 2q \cdot t \cdot \delta_\mathsf{G}$ for a distinguisher D of size s. Then we can construct a distinguisher D' of size at most $s + q \cdot |C^\mathsf{G}|$ (which is equal to $s_\mathsf{G} - 2q \cdot t \cdot |\mathsf{G}|$), that distinguishes a q-tuple of samples of $\mathsf{G}^{2t}(U_n)$ from a q-tuple of samples of U_{2tn} with a distinguishing advantage $> 2q \cdot t \cdot \delta_\mathsf{G}$. Using a standard hybrid argument this distinguisher yields another distinguisher D'' that distinguishes between a single sample of $\mathsf{G}^{2t}(U_n)$ from a single sample of U_{2tn} with a distinguishing advantage $> 2 \cdot t \cdot \delta_\mathsf{G}$. This contradicts Lemma 1.

Now we describe our distinguisher D'. D' gets as input a q-tuple $(a_1, a_2 \ldots a_q)$ which has samples either from $\mathsf{G}^{2t}(U_n)$ or from U_{2tn}. D' internally executes D and answers the oracle queries of D by executing C_1. However it uses a_i instead of generating values using G^{2t}. More specifically, it uses a fresh value of a_i for every query, except for repeat queries. In case of a repeat query it responds with the value that was returned previously (for the query being repeated). If the input tuple consists of samples from $\mathsf{G}^{2t}(U_n)$, then the distribution corresponds to the circuit C_1. On the other hand if the samples are from U_{2tn}, then the distribution corresponds to the circuit C_2. Hence the success of D in distinguishing between the two cases directly translates to the success of D' in distinguishing q-tuple of $\mathsf{G}^{2t}(U_n)$ from q-tuple of U_{2tn}. Note that the size of our distinguisher D' is larger than the size of D by at most $q \cdot |C^\mathsf{G}|$. ■

[-] **Hybrid $\mathcal{H}_3$:** This hybrid is the same as the previous hybrid except that we use one random function $\mathcal{R}_{t,2nt}$ instead of two functions $\mathcal{R}_{n,2nt}$ and $\mathcal{R}_{2t,n}$. More specifically, $\mathsf{C}_3(k, x) = h_2(\mathcal{R}_{n,2nt}(h_1(k_0, x)), x)$. For any machine D^{C_3} of size s and that makes q queries to C_3 let $p_3^D = \Pr[D^{\mathsf{C}_3(\cdot)} = 1]$. We now show the following lemma:

Lemma 4 *For every adversarial q-query distinguisher D we have (unconditionally) that $|p_3^D - p_2^D| \leq \frac{q^2}{2^n}$.*

Proof. Observe that C_2 consists of two nested random functions $f(\cdot) = \mathcal{R}_{n,2nt}(\mathcal{R}_{2t,n}(\cdot))$ and on the other hand C_3 consists of one random function $g(\cdot) = \mathcal{R}_{2t,2nt}(\cdot)$. Further, note that every time C_2 (resp., C_3) is called $f(\cdot)$ (resp., $g(\cdot)$) is executed exactly once. Hence, the distinguishing advantage of an unbounded q-query distinguisher D can be bounded by the distinguishing advantage of an unbounded distinguisher (that makes q queries) in distinguishing between $f(\cdot)$ and $g(\cdot)$.

Without loss of generality we assume that all the queries of the distinguisher are distinct. Let E be the event that the q queries (all distinct among themselves) of the distinguisher are such that all of the queries to the internal random function $\mathcal{R}_{2t,n}(\cdot)$ in $f(\cdot)$ are distinct. Observe that, conditioned on the event E, the distributions generated by $f(\cdot)$ and $g(\cdot)$ are the same. Hence the distinguishing advantage of an unbounded distinguisher (that makes q queries), in distinguishing between $f(\cdot)$ and $g(\cdot)$, can be upper bounded by the probability of event E failing to happen. This corresponds to the probability that q uniformly random values are such that there is a collision among two values. This value is $\frac{\binom{q}{2}}{2^n}$ which is upper bounded by $\frac{q^2}{2^n}$. ■

[-] **Hybrid $\mathcal{H}_4$:** This hybrid corresponds to a random function. More specifically, $\mathsf{C}_4(k, x) = \mathcal{R}_{n,m}(x)$. For any machine D^{C_4} of size s that makes q queries to C_4, let $p_4^D = \Pr[D^{\mathsf{C}_4(\cdot)} = 1]$.

Lemma 5 *For every adversarial q-query distinguisher D we (unconditionally) have that* $|p_4^D - p_3^D| \leq \frac{q^2}{2^{t\alpha}} + q^2 \cdot 2^{t\alpha}(\delta_1 + q \cdot \delta_2) + q \cdot \delta_2$.

Proof. We prove this lemma using a step ladder approach. For an adaptive distinguisher D let E_i^D be the event such that D succeeds in making adaptive queries $x_1, x_2 \ldots x_i$ such that there exists a subset $\mathcal{I} \subseteq [i]$ of size $|\mathcal{I}| = \alpha + 1$ such that $h_1(k_0, x_j) = h_1(k_0, x_k)$ for all $j, k \in \mathcal{I}$. Intuitively speaking E_i^D is the event that D is able to force an $\alpha+1$-wise collision on the output of the inner hash function in the i queries that it makes. At this point we claim the following lemma and prove it separately. This lemma will be used crucially in the rest of the analysis.

Lemma 6 *For every adversarial i-query distinguisher D we (unconditionally) have that* $|\mathsf{Pr}[D^{\mathsf{C}_3(\cdot)} = 1|\neg E_i^D] - \mathsf{Pr}[D^{\mathsf{C}_4(\cdot)} = 1|\neg E_i^D]| \leq i \cdot \delta_2$.

Proof. We start by stressing that the distinguisher here is only allowed to make i queries and all the probabilities considered in this proof are for the setting where we condition on $\neg E_i^D$.

Consider a sequence of $i+1$ hybrids – $Y_0, Y_1 \ldots Y_i$. In the hybrid Y_j (for $0 \leq l \leq i$) the adaptive query x_j for $j \in \{0, \ldots, i\}$ is answered as follows:

- *If* $j < l$ then return $\mathcal{R}_{n,m}(x_j)$.
- *Else* return $h_2(\mathcal{R}_{n,2tn}(h_1(k_0, x_j)), x_j)$.

We will next argue that for every $l \in \{0, 1, \ldots i-1\}$ the statistical difference between the hybrids Y_l and Y_{l+1}, when restricted to the case $\neg E_i^D$, is bounded by δ_2. This directly implies the claimed lemma.

Now we will argue that any adaptive distinguisher D distinguishing between Y_l and Y_{l+1} with a probability greater that δ_2 can be used to construct a distinguisher D' that distinguishes the $(\alpha+1)$-wise independent hash function $h_2(k_1, \cdot)$ (for a random seed k_1) from $\mathcal{S}_{n,m}(\cdot)$ with probability at least δ_2 when making at most α adaptive queries to $\mathcal{O}(\cdot)$, where $\mathcal{O}(\cdot)$ is either $h_2(k_1, \cdot)$ (for a random seed k_1) or $\mathcal{S}_{n,m}(\cdot)$; here $\mathcal{S}_{n,m}(\cdot)$ is just a random oracle. D' internally executes D which makes i adaptive queries on inputs $x_1, \ldots x_i$. D' provides answers to the query x_j for $j \in \{0, \ldots, i\}$ as follows:

- *If* $j < l$ then return $\mathcal{R}_{n,m}(x_j)$.
- *If* $j = l$ then return $\mathcal{S}_{n,m}(x_j)$.
- *Else* return $h_2(\mathcal{R}_{n,2tn}(h_1(k_0, x_j)), x_j)$.

Observe that the view of D when $\mathcal{O}$ is $h_2(k_1, \cdot)$ corresponds to the hybrid Y_l. On the other hand the view of D when $\mathcal{O}$ is $\mathcal{S}_{n,m}(\cdot)$ corresponds to the hybrid Y_{l+1}. Note that both $\mathcal{R}$ and $\mathcal{S}$ are random oracles with identical output distributions Finally, note that, since we are conditioning on the event that $\neg E_i^D$, it follows that D' makes at most α adaptive queries to $\mathcal{O}(\cdot)$. Hence our claim follows. ■

We now complete the proof of Lemma 5. We start by evaluating the probability $\Pr[E_{i+1}^D|\neg E_i^D]$. First, consider the case where an adaptive distinguisher D is not given any of the responses. Now, note that the probability, for the $i+1^{th}$ query x_{i+1} made by adaptive D, to be such that for a particular subset $\mathcal{I} \subseteq [i]$ of size $|\mathcal{I}| = \alpha+1$, $h_1(k_0, x_j) = h_1(k_0, x_k)$ for all $j, k \in \mathcal{I}$, is $\frac{1}{2^{2 \cdot t \cdot \alpha}} + \delta_1$. By Lemma 6 given $\neg E_i^D$ we have that the statistical difference between the responses actually provided and uniformly random values is $i \cdot \delta_2$. Therefore, we can claim that given the responses, the success probability of D can increase by at most $i \cdot \delta_2$. Hence we have that the probability that the $i+1^{th}$ query made by adaptive D (when it is actually provided with the responses) such that for a particular subset $\mathcal{I} \subseteq [i]$ of size $|\mathcal{I}| = \alpha+1$, $h_1(k_0, x_j) = h_1(k_0, x_k)$ for all $j, k \in \mathcal{I}$, is at most $\frac{1}{2^{2t \cdot \alpha}} + \delta_1 + i\delta_2$. Taking union bound over all $\frac{(i+1)^{\alpha+1}}{(\alpha+1)!}$ possible $\alpha+1$-element subsets of the $i+1$ element set we get that:

$$\begin{aligned}
\Pr[E_{i+1}^D|\neg E_i^D] &\leq \frac{(i+1)^{\alpha+1}}{(\alpha+1)!} \cdot \left(\frac{1}{2^{2t\alpha}} + \delta_1 + i\delta_2\right) \\
&\leq 2^{(\alpha+1) \cdot \log{(i+1)}} \cdot \left(\frac{1}{2^{2t\alpha}} + \delta_1 + i\delta_2\right) \qquad (1) \\
&\leq 2^{t\alpha+t} \cdot \left(\frac{1}{2^{2t\alpha}} + \delta_1 + q\delta_2\right) = \left(\frac{q}{2^{t\alpha}} + 2^{t\alpha} \cdot q \cdot (\delta_1 + q\delta_2)\right)
\end{aligned}$$

Next,

$$
\begin{aligned}
\Pr[\neg E_q^D] &= \Pr[\neg E_q^D | \neg E_{q-1}^D] \Pr[\neg E_{q-1}^D] = (1 - \Pr[E_q^D | \neg E_{q-1}^D]) \Pr[\neg E_{q-1}^D] \\
&= \prod_{i=0}^{q-1} (1 - \Pr[E_{i+1}^D | \neg E_i^D]) \geq \prod_{i=0}^{q-1} (1 - \frac{q}{2^{t\alpha}} - 2^{t\alpha} q(\delta_1 + q \cdot \delta_2)) \\
&\geq (1 - \frac{q}{2^{t\alpha}} - 2^{t\alpha} q(\delta_1 + q \cdot \delta_2))^q \geq 1 - \frac{q^2}{2^{t\alpha}} - q^2 \cdot 2^{t\alpha}(\delta_1 + q \cdot \delta_2)
\end{aligned} \tag{2}
$$

Finally,

$$
\begin{aligned}
&|\mathsf{Pr}[D^{\mathsf{C}_3(\cdot)} = 1] - \mathsf{Pr}[D^{\mathsf{C}_4(\cdot)} = 1]| \\
&\qquad \leq |\mathsf{Pr}[D^{\mathsf{C}_3(\cdot)} = 1 | \neg E_q^D] \Pr[\neg E_q^D] + \mathsf{Pr}[D^{\mathsf{C}_3(\cdot)} = 1 | E_q^D] \Pr[E_q^D] \\
&\qquad - \mathsf{Pr}[D^{\mathsf{C}_4(\cdot)} = 1 | \neg E_q^D] \Pr[\neg E_q^D] - \mathsf{Pr}[D^{\mathsf{C}_4(\cdot)} = 1 | E_q^D] \Pr[E_q^D]| \\
&\qquad \leq (|\mathsf{Pr}[D^{\mathsf{C}_3(\cdot)} = 1 | \neg E_q^D] - \mathsf{Pr}[D^{\mathsf{C}_4(\cdot)} = 1 | \neg E_q^D]|) \Pr[\neg E_q^D] \\
&\qquad + (|\mathsf{Pr}[D^{\mathsf{C}_3(\cdot)} = 1 | E_q^D] - \mathsf{Pr}[D^{\mathsf{C}_4(\cdot)} = 1 | E_q^D]|) \Pr[E_q^D] \\
&\qquad \leq |\mathsf{Pr}[D^{\mathsf{C}_3(\cdot)} = 1 | \neg E_q^D] - \mathsf{Pr}[D^{\mathsf{C}_4(\cdot)} = 1 | \neg E_q^D]| + \Pr[E_q^D] \\
&\qquad \leq q \cdot \delta_2 + \frac{q^2}{2^{t\alpha}} + q^2 \cdot 2^{t\alpha}(\delta_1 + q \cdot \delta_2)
\end{aligned} \tag{3}
$$

This completes the proof of the claimed lemma. ■

The proof of Theorem 1 follows from Lemmas 2,3,4 and 5.

5 Conclusion

Pseudorandom functions (PRF) are one of the most fundamental primitives in cryptography, both from a theoretical and a practical standpoint. However unfortunately, many of the known black-box constructions from PRGs are inefficient. Recently, Jain, Pietrzak, and Tentes [13] gave a hardness-preserving construction of a PRF that makes only $O(\log q)$ calls to the underlying PRG when $q = 2^{n^\epsilon}$ and $\epsilon \geq \frac{1}{2}$. This dramatically improves upon the efficiency of the GGM construction. However, they explicitly left open the question of whether such constructions exist when $\epsilon < \frac{1}{2}$. In this work, we give constructions of PRFs that make only $O(\log q)$ calls to the underlying PRG even when $q = 2^{n^\epsilon}$ for $0 < \epsilon < \frac{1}{2}$.

References

[1] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.

[2] Itay Berman and Iftach Haitner. From non-adaptive to adaptive pseudorandom functions. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 357–368, 2012.

[3] Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness preserving reductions via cuckoo hashing. *IACR Cryptology ePrint Archive*, 2012:722, 2012.

[4] Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness preserving reductions via cuckoo hashing. In *TCC*, pages 40–59, 2013.

[5] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 112–117, 1982.

[6] Larry Carter and Mark N. Wegman. Universal classes of hash functions (extended abstract). In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 106–112, 1977.

[7] Nishanth Chandran and Sanjam Garg. Hardness preserving constructions of pseudorandom functions, revisited. *IACR Cryptology ePrint Archive*, 2012:616, 2012.

[8] Chongwon Cho, Chen-Kuei Lee, and Rafail Ostrovsky. Equivalence of uniform key agreement and composition insecurity. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 447–464, 2010.

[9] Oded Goldreich. Towards a theory of software protection. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 426–439, 1986.

[10] Oded Goldreich. Two remarks concerning the goldwasser-micali-rivest signature scheme. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 104–110, 1986.

[11] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 276–288, 1984.

[12] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[13] Abhishek Jain, Krzysztof Pietrzak, and Aris Tentes. Hardness preserving constructions of pseudorandom functions. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 369–382, 2012.

[14] Kaoru Kurosawa, Thomas Johansson, and Douglas R. Stinson. Almost k-wise independent sample spaces and their cryptologic applications. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 409–421, 1997.

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

[16] Michael Luby. *Pseudorandomness and cryptographic applications*. Princeton computer science notes. Princeton University Press, 1996.

[17] Michael Luby and Charles Rackoff. A study of password security. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 392–397, 1987.

[18] Ueli M. Maurer. Indistinguishability of random systems. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 110–132, 2002.

[19] Steven Myers. Black-box composition does not imply adaptive security. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 189–206, 2004.

[20] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[21] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of psuedo-random functions. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 170–181, 1995.

[22] Krzysztof Pietrzak. Composition does not imply adaptive security. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 55–65, 2005.

[23] Krzysztof Pietrzak. Composition implies adaptive security in minicrypt. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 328–338, 2006.

[24] Mark N. Wegman and Larry Carter. New classes and applications of hash functions. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 175–182, 1979.

[25] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.