Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 182 (2012)

Hardness-Preserving Reductions via Cuckoo Hashing*

Itay Berman[†§] Iftach Haitner[‡§] Ilan Komargodski[¶] Moni Naor[¶]

October 20, 2015

Abstract

The focus of this work is *hardness-preserving* transformations of somewhat limited pseudorandom functions families (PRFs) into ones with more versatile characteristics. Consider the problem of *domain extension* of pseudorandom functions: given a PRF that takes as input elements of some domain $\mathcal{U}$, we would like to come up with a PRF over a larger domain. Can we do it with little work and without significantly impacting the security of the system? One approach is to first hash the larger domain into the smaller one and then apply the original PRF. Such a reduction, however, is vulnerable to a "birthday attack": after $\sqrt{|\mathcal{U}|}$ queries to the resulting PRF, a collision (i.e., two distinct inputs having the same hash value) is very likely to occur. As a consequence, the resulting PRF is *insecure* against an attacker making this number of queries.

In this work we show how to go beyond the aforementioned birthday attack barrier by replacing the above simple hashing approach with a variant of *cuckoo hashing*, a hashing paradigm that resolves collisions in a table by using two hash functions and two tables, cleverly assigning each element to one of the two tables. We use this approach to obtain: (i) a domain extension method that requires *just two calls* to the original PRF, can withstand as many queries as the original domain size, and has a distinguishing probability that is exponentially small in the amount of non-cryptographic work; and (ii) a *security-preserving* reduction from non-adaptive to adaptive PRFs.

1 Introduction

The focus of this work is *hardness-preserving* transformations of somewhat limited pseudorandom functions families (PRFs) into ones with more versatile characteristics. Examples of somewhat limited such families include those with *small* domain or those that can withstand only *non-adaptive* (also known as static) attacks, in which the attacker chooses its queries ahead of time, before seeing any of the answers. In contrast, less limited families might have *large* domain or be

*A preliminary version appeared in the proceedings of TCC 2013 [8].

[†]MIT Computer Science and Artificial Intelligence Laboratory (CSAIL). Email: itayberm@mit.edu. Most of this work was done while the author was in the School of Computer Science, Tel Aviv University.

[‡]School of Computer Science, Tel Aviv University. Email: iftachh@cs.tau.ac.il.

[§]Research supported in part by Check Point Institute for Information Security and the (I-CORE) program (Center No. 4/11) of the Planning and Budgeting Committee.

[¶]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. Email: {ilan.komargodski,moni.naor}@weizmann.ac.il. Research supported in part by a grant from the I-CORE Program (Center No. 4/11) of the Planning and Budgeting Committee and the Israel Science Foundation. Moni Naor is the incumbent of the Judith Kleeman Professorial Chair.

ISSN 1433-8092

secure against *adaptive* (dynamic) attacks, in which the attacker's queries might be chosen as a function of all previous answers.

A common paradigm, first suggested by Levin [29, §5.4], for increasing the usability and security of a PRF, is to "hash" the inputs into a smaller domain *before* applying the PRF. This approach was originally suggested in order to achieve "PRF domain extension" (using a short, e.g., fixed, input length PRF to get a variable-length PRF); more recently, it was used to transform non-adaptive PRFs into adaptive ones [7]. Such reductions, however, are vulnerable to the following "birthday attack": after $\sqrt{|\mathcal{U}|}$ queries to the resulting PRF, where $\mathcal{U}$ is the hash function range, a collision (i.e., two distinct inputs having the same hash value) is very likely to occur. Such collisions are an obstacle to the indistinguishability of the PRF, since in a random function we either do not expect to see a collision at all (if the range is large enough) or expect to see fewer collisions. Hence, the resulting PRF is *insecure* against an attacker making this number of queries.

In this work we study variants of the above hashing approach to go beyond the birthday attack barrier. In a high-level, our approach, which can be traced back to Siegel [51], is based on applying a dictionary data structure[1], in which the locations accessed in the search of an element are determined by its value and some fixed random string (i.e., the same string is used for all elements), and not on values seen during the search. Now to do the conversion to domain extension we assign random values to all locations (by the underlying PRF). We think of the large domain as the universe from which the elements of the dictionary are taken. The resulting value of the extended function at point x will be some (simple) function of the values assigned to the locations accessed during the search for x. For instance, one can view Levin's construction above as an instance of this framework, where the fixed random string describes a hash function from large domain to a smaller-size set $\mathcal{U}$, and the PRF, whose domain is $\mathcal{U}$, assigns random values for $|\mathcal{U}|$ locations. The distinguishing probability of the resulting scheme is the distinguishing probability of the underlying PRF *plus* the probability of failure of the dictionary (which in Levin's construction is determined by the "birthday paradox"). The cost of the extension is related to the worst case search time of the dictionary (which in Levin's construction is a single invocation of the hash function).

We focus on constructions based on *cuckoo hashing*: a hashing paradigm typically used for resolving hash collisions in a table by using two hash functions and two tables, assigning each element to one of the two tables, and enabling lookup using only two queries (see Section 1.2). We use this paradigm to present a new PRF domain extension method that requires *just two calls* to the original PRF, can withstand as many queries as the original domain size, and has a distinguishing probability that is exponentially small in the amount of non-cryptographic work. We also obtain a *security-preserving reduction* from non-adaptive to adaptive PRFs, an improvement upon the recent result of Berman and Haitner [7].

Before stating our results, we discuss in greater detail pseudorandom functions and cuckoo hashing.

1.1 Pseudorandom Functions

Pseudorandom function families (PRFs), introduced by Goldreich, Goldwasser, and Micali [21], are function families that cannot be distinguished from a family of *truly* random functions by an efficient distinguisher given an oracle access to a random member of the family. PRFs have an

[1] In this context a dictionary is a data structure used for maintaining a set of elements while supporting membership queries.

extremely important role in cryptography, allowing parties who share a common secret key to send secure messages, identify themselves, and authenticate messages [20, 30]. They have many other applications as well, and can be used in just about any setting that requires a random function provided as a black-box [5, 10, 13, 19, 31, 41]. Different PRF constructions, whose security is based on different hardness assumptions, are known in the literature. The construction most relevant to this work is the one of [21], hereafter the $\mathcal{GGM}$ construction, which uses a length-doubling pseudorandom generator (and thus can be based on the existence of one-way functions [23]).

We use the following definitions: an efficiently computable function family ensemble $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ is a (q, t, ε)-PRF, if (for large enough n) a $q(n)$-query oracle-aided algorithm (distinguisher) of running time $t(n)$, getting access to a random function from the family, distinguishes between $\mathcal{F}_n$ and the family of all functions (with the same input/output domains), with probability at most $\varepsilon(n)$. $\mathcal{F}$ is a *non-adaptive* (q, t, ε)-PRF if it is only required to be secure against non-adaptive distinguishers (i.e., ones that prepare all their queries in advance). Finally, $\mathcal{F}$ is a t-PRF if q is only limited by t and $\varepsilon = 1/t$.

We also make use of the information-theoretic analog of a t-PRF, known as a t-wise independent family, that is formally defined in Definition 2.5.

1.2 Cuckoo Hashing and Many-wise Independent Hash Function

Cuckoo hashing, introduced by Pagh and Rodler [43], is an efficient technique for constructing dynamic dictionaries. Such data structures are used to maintain a set of elements, while supporting membership queries as well as insertions and deletions of elements. Cuckoo hashing maintains such a dynamic dictionary by keeping two tables of size only slightly larger than the number of elements to be inserted, and two hash functions mapping the elements into cells of those tables. It then applies a clever algorithm for placing at most a single element in each cell. Each membership query requires just two memory access (in the worst case) and they are determined by the hash functions. Many variants of cuckoo hashing have been proposed since its introduction, and extensive literature has been devoted to its analysis (cf., [17, 14, 28, 18, 2]).

Pagh and Pagh [42] used ideas in the spirit of cuckoo hashing to construct efficient many-wise independent hash functions. Let $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{F}$ be function families from $\mathcal{D}$ to $\mathcal{S}$, from $\mathcal{D}$ to $\mathcal{R}$ and from $\mathcal{S}$ to $\mathcal{R}$ respectively, where $\mathcal{R}$ is a group with operation $\oplus$. Define the function family $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F})$ from $\mathcal{D}$ to $\mathcal{R}$ as

$$\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F}) = (\mathcal{F} \circ \mathcal{H}) \oplus (\mathcal{F} \circ \mathcal{H}) \oplus \mathcal{G}, \tag{1}$$

where $\mathcal{F}_1 \circ \mathcal{F}_2$, for function families $\mathcal{F}_1$ and $\mathcal{F}_2$, is the function family whose members are the elements of $\mathcal{F}_1 \times \mathcal{F}_2$ and $(f_1, f_2)(x)$ is defined by $f_1(f_2(x))$ ($\mathcal{F}_1 \oplus \mathcal{F}_2$ is analogously defined). In other words, given $f_1, f_2 \in \mathcal{F}, h_1, h_2 \in \mathcal{H}$ and $g \in \mathcal{G}$, design a function $\mathcal{PP}_{f_1,f_2,h_1,h_2,g}(x) = f_1(h_1(x)) \oplus f_2(h_2(x)) \oplus g(x)$. Pagh and Pagh [42] showed that when the families $\mathcal{H}$ and $\mathcal{G}$ are of "high enough" independence, that is, roughly $(c \cdot \log|\mathcal{S}|)$-wise independent, then the family $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \Pi)$ is $O(|\mathcal{S}|^{-c})$-indistinguishable from random by a $|\mathcal{S}|$-query, *non-adaptive* distinguisher, where Π is the set of all functions from $\mathcal{S}$ to $\mathcal{R}$. Note that the security of the resulting family goes well beyond the birthday attack barrier: it is indistinguishable from random by an attacker making $|\mathcal{S}| \gg \sqrt{|\mathcal{S}|}$ queries.

Aumüller et al. [4] (building on the work of Dietzfelbinger and Woelfel [15]) strengthen the result of [42] by using more sophisticated hash functions $\mathcal{H}$ and $\mathcal{G}$ (rather than the $O(\log|\mathcal{S}|)$-wise

independent that [42] used). Specifically, for a given $s \geq 0$, Aumüller et al. [4] constructed a function family $\mathcal{ADW}_s(\mathcal{H}, \mathcal{G}, \Pi)$ that is $O(|\mathcal{S}|^{-(s+1)})$-indistinguishable from random by a $|\mathcal{S}|$-query, *non-adaptive* distinguisher, where Π is the set of all functions from $\mathcal{S}$ to $\mathcal{R}$.[2] The idea to use more sophisticated hash functions, in the sense that they require less combinatorial work, already appeared in previous works, e.g., the work of Arbitman et al. [2, §5.4].

In Section 3 we take the above results a step further, showing that they hold also for *adaptive* distinguishers.[3] Our approach for this transformation has many predecessors. For instance, the work of Naor and Reingold [38], and of Jetchev et al. [26]. Furthermore, it turns out that by using the above function family with a pseudorandom function $\mathcal{F}$, namely the family $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F})$ (or $\mathcal{ADW}_s(\mathcal{H}, \mathcal{G}, \mathcal{F})$), we get a pseudorandom function that is superior to $\mathcal{F}$ (the actual properties of $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F})$ are determined by the properties of $\mathcal{F}$ and the choice of $\mathcal{H}$ and $\mathcal{G}$). This understanding is the main conceptual contribution of this paper, and the basis for the results presented below.

We note that the works of Pagh and Pagh [42], Dietzfelbinger and Woelfel [15], and Aumüller et al. [4] have gone almost unnoticed in the cryptography literature so far.[4] In this work we apply, in a black-box manner, the analysis of [42] and of [4] in cryptographic settings.

1.3 Our Results

We use a construction inspired by cuckoo hashing to improve upon two PRF reductions: PRF domain extension and non-adaptive to adaptive PRF.

1.3.1 PRF Domain Extension

PRF domain extensions use PRFs with "small" domain size to construct PRFs with larger (or even unlimited) domain size. These extensions reduce the cost of a single invocation of the PRF and increase its usability. Domain extension methods are typically measured by the security of the resulting PRFs, and by the number of calls the resulting PRF makes to the underlying PRF.

Among the known domain extension techniques are the MAC-based constructions, such as CBC-MAC and PMAC (a survey on their security can be found in [37]). The number of calls made by these constructions to the underlying (small domain) PRF can be as small as two. Assuming that the underlying PRF is a random function over $\{0,1\}^n$, then the resulting family is $(q, \infty, O(q^2/2^n))$-PRF (i.e., the ∞ in the second parameter means that the distinguisher's running time is unlimited). A second technique is the Feistel or Beneš transformations (e.g., [1, 44, 45], a survey of which can be found in [46]). The Beneš based construction makes 8 calls to the underlying PRF and is $(q, \infty, O(q/2^n))$-PRF, whereas a 5-round Feistel based construction (which makes 5 calls to the underlying PRF) is $(q, \infty, O(q/2^n))$-PRF.

Our cuckoo hashing based function family (see below) is $(q, \infty, O(q/2^n))$-PRF and makes only two calls to the underlying PRF. Moreover, our construction can extend the domain size to any

[2] The $\mathcal{ADW}$'s function family is in fact more complicated than the above simplified description. See Section 6 for the formal definition.

[3] Note that in some cases an adaptive adversary is a more powerful distinguisher than a non-adaptive one. For example, when a trying to distinguish between a truly random function and a random involution (permutations where the cycle length is at most 2). There exists an adaptive distinguisher that will succeed with very high probability by asking two queries while any non-adaptive distinguisher will fail with very high probability (see [27, 40]).

[4] Pagh and Pagh [42] did notice this connection, and in particular mentioned the connection of their work to that of Bellare et al. [6].

poly(n) length, unlike the aforementioned constructions, which only double the domain size.[5]

Theorem 1.1 (informal)**.** *Let $k \leq n$, let $\mathcal{H}$ and $\mathcal{G}$ be efficient k-wise independent function families mapping strings of length $\ell(n)$ to strings of length n, and let Π be the family of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. Then the family $\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi)$, mapping strings of length $\ell(n)$ to strings of length n, is a $(q,\infty,q/2^{\Omega(k)})$-PRF, for $q \leq 2^{n-2}$.*

For $k = \Theta(n)$, Theorem 1.1 yields a domain extension that is $(q,\infty,q/2^n)$-PRF, and makes only two calls to the underlying PRF. Replacing in the above construction the function family $\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi)$ with the family $\mathcal{ADW}_s(\mathcal{H},\mathcal{G},\Pi)$ yields a more versatile domain extension that offers a tradeoff between the number of calls to the PRF and the independence required for the hash functions. For details, see Section 6.2.

PRG to PRF reductions. Theorem 1.1 can also be used to get a hardness-preserving construction of PRFs from pseudorandom generators (PRG) in settings where there is a non-trivial bound on the number of queries to the PRF. Jain et al. [25], who were the first to propose this goal, noted that one can realize it using a domain extension constructions. Thus, we apply Theorem 1.1 to get constructions of PRFs from PRGs which improves some of the parameters of Jain et al. [25], but require longer keys. See Appendix B for details.

1.3.2 From Non-Adaptive to Adaptive PRF

Adaptive PRFs can be constructed from non-adaptive ones using general techniques such as using the PRG-based construction of Goldreich et al. [21] or the *synthesizer* based construction of Naor and Reingold [39]. These constructions, however, make (roughly) n calls to the underlying non-adaptive PRF (where n is the input length). Recently, Berman and Haitner [7] showed how to perform this security uplift at a much lower cost: the adaptive PRF makes only a *single* call to the non-adaptive PRF. Their construction, however, incurs a significant degradation in the security: assuming the underlying function is a non-adaptive t-PRF, then the resulting function is an (adaptive) $O(t^{1/3})$-PRF. The reason for this degradation is the birthday attack we mentioned earlier.

We present a reduction from non-adaptive to adaptive PRFs that *preserves* the security of the non-adaptive PRF. The resulting adaptive PRF makes only two calls to the underlying non-adaptive one.

Theorem 1.2 (informal)**.** *Let t be a polynomial-time computable integer function, let $\mathcal{H} = \{\mathcal{H}_n\colon \{0,1\}^n \mapsto [4t(n)]_{\{0,1\}^n}\}_{n\in\mathbb{N}}$ (where $[4t(n)]_{\{0,1\}^n}$ are the first $4t(n)$ elements of $\{0,1\}^n$) and $\mathcal{G} = \{\mathcal{G}_n\colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be efficient $O(\log t(n))$-wise independent function families, and let $\mathcal{F}$ be a length-preserving non-adaptive t-PRF. Then $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ is a length-preserving $(t/4)$-PRF.*

Also in this case, replacing the function family $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ with the family $\mathcal{ADW}_s(\mathcal{H},\mathcal{G},\mathcal{F})$, yields a more versatile non-adaptive to adaptive transformation that offers a trade-off between the number of calls to the PRF and the requisite independence. See Section 6.3 for details.

[5] Of course, one can use these constructions to extend the domain to any poly(n) length by a recursive construction. This, however, will increase the number of calls to the underlying PRF by a polynomial factor.

1.4 More Related Work

Bellare et al. [6] introduced a paradigm for using PRFs in the symmetric-key settings that, in retrospect, is similar to cuckoo hashing. Assume that two parties, who share a secret function f would like to use it for (shared-key) encryption. The 'textbook' (stateless) solution calls for the sender to choose r at random and send $(r, f(r) \oplus M)$ to the receiver, where M is the message to be encrypted. This proposal breaks down if the sender chooses the same r twice (in two different sessions with different messages). Thus, the scheme is subject to the birthday attack and the length parameters should be chosen accordingly. This requires the underlying function to have a large domain. Instead, [6] suggested choosing $t > 1$ values at random, and sending $(r_1, \ldots, r_t, f(r_1) \oplus \cdots \oplus f(r_t) \oplus M)$. They were able to show much better security than the single r case. They also showed a similar result for message authentication. Our domain extension results (see Section 4) improve upon the results of [6].

The problem of transforming a scheme that is only resilient to non-adaptive attack into one that is resilient to adaptive attacks has received quite a lot of attention in the context of pseudo-random permutations (or block ciphers). Maurer and Pietrzak [33] showed that, given a family of permutations that are information-theoretic secure against non-adaptive attacks, if two members of this family are independently composed, then the resulting permutation is also secure against adaptive attacks (see [33] for the exact formulation). Pietrzak [47] showed, however, that this is not necessarily the case for permutations that are random-looking under a computational assumption (see also [36, 48]), reminding us that translating information-theoretic results to the computational realm is a tricky business.

Paper Organization

Basic notations and formal definitions are given in Section 2. In Section 3 we formally define the hashing paradigm of Pagh and Pagh [42] and show how to extend their result to hold against adaptive adversaries. Our domain extension reduction based on [42] is described in Section 4, and the improved non-adaptive to adaptive reduction, also based on [42], is described in Section 5. In Section 6 we present the more advanced (and more complex) hashing paradigm of Aumüller et al. [4], and use it to obtain a more versatile version of the above reductions. Some possible directions for future research are discussed in Section 7.

2 Preliminaries

2.1 Notations

All logarithms considered here are in base two. We use calligraphic letters to denote sets, uppercase for random variables, and lowercase for values. Let '||' denote string concatenation. For an integer t, let $[t] = \{1, \ldots, t\}$. For a set $\mathcal{S}$ and integer t, let $\mathcal{S}^{\leq t} = \{\overline{s} \in \mathcal{S}^* \colon\ |\overline{s}| \leq t \ \wedge\ \overline{s}[i] \neq \overline{s}[j]\ \ \forall i \neq j \in [|\overline{s}|]\}$, and let $[t]_{\mathcal{S}}$ be the first t elements, in increasing lexicographic order, of $\mathcal{S}$ (equal to $\mathcal{S}$ in case $|\mathcal{S}| < t$). For sets $\mathcal{U}$ and $\mathcal{V}$, let $\Pi_{\mathcal{U} \mapsto \mathcal{V}}$ stands for the set of all functions from $\mathcal{U}$ to $\mathcal{V}$, and for integers n and ℓ, let $\Pi_{n,\ell} = \Pi_{\{0,1\}^n \mapsto \{0,1\}^\ell}$.

Let poly denote the set all polynomials, and let PPTM be abbreviation for probabilistic (strictly) polynomial-time Turing machine. For $s \in \mathbb{N}$ and $t, q \colon \mathbb{N} \mapsto \mathbb{N}$, we say that D is a t-time q-query s-oracle-aided algorithm if, when invoked on input of length n, D runs in time $t(n)$ and makes at

most $q(n)$ queries to each of its s oracles. Given a random variable X, we write $X(x)$ to denote $\Pr[X = x]$, and write $x \leftarrow X$ to indicate that x is selected according to X. Similarly, given a finite set $\mathcal{S}$, we let $s \leftarrow \mathcal{S}$ denote that s is selected according to the uniform distribution on $\mathcal{S}$. The *statistical distance* of two distributions P and Q over a finite set $\mathcal{U}$, denoted as $\mathrm{SD}(P, Q)$, is defined as $\max_{\mathcal{S}\subseteq\mathcal{U}} |P(\mathcal{S}) - Q(\mathcal{S})| = \frac{1}{2}\sum_{u\in\mathcal{U}} |P(u) - Q(u)|$.

2.2 Pseudorandom Generators

Definition 2.1 (Pseudorandom Generators)**.** *A polynomial-time function $G\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is (t,ε)-*PRG*, if $\ell(n) > n$ for every $n \in \mathbb{N}$ (G stretches the input), and*

$$\left|\Pr_{x\leftarrow\{0,1\}^n}[\mathsf{D}(G(x)) = 1] - \Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathsf{D}(y) = 1]\right| \leq \varepsilon(n)$$

for every algorithm (distinguisher) D of running time $t(n)$ and large enough n. A $(t, 1/t))$-PRG is called a t-PRG. If $\ell(n) = 2n$, we say that G is length-doubling.

2.3 Function Families

2.3.1 Operating on Function Families

We consider two natural operations on function families.

Definition 2.2 (composition of function families)**.** *Let $\mathcal{F}^1\colon \mathcal{D}^1 \mapsto \mathcal{R}^1$ and $\mathcal{F}^2\colon \mathcal{D}^2 \mapsto \mathcal{R}^2$ be two function families with $\mathcal{R}^1 \subseteq \mathcal{D}^2$. The* composition *of $\mathcal{F}^1$ with $\mathcal{F}^2$, denoted $\mathcal{F}^2 \circ \mathcal{F}^1$, is the function family $\{(f_2, f_1) \in \mathcal{F}^2 \times \mathcal{F}^1\}$, where $(f_2, f_1)(x) := f_2(f_1(x))$.*

Definition 2.3 (group operation of function families)**.** *Let $\mathcal{F}^1\colon \mathcal{D} \mapsto \mathcal{R}^1$ and $\mathcal{F}^2\colon \mathcal{D} \mapsto \mathcal{R}^2$ be two function families with $\mathcal{R}^1, \mathcal{R}^2 \subseteq \mathcal{R}$, where $\mathcal{R}$ is a group with operation $\oplus$. The* group operation *of $\mathcal{F}^1$ with $\mathcal{F}^2$, denoted $\mathcal{F}^2 \bigoplus \mathcal{F}^1$, is the function family $\{(f_2, f_1) \in \mathcal{F}^2 \times \mathcal{F}^1\}$, where $(f_2, f_1)(x) := f_2(x) \oplus f_1(x)$.*

In all of our applications, the group $\mathcal{R}$ from the above definition will simply be $\{0,1\}^n$ for some $n \in \mathbb{N}$, with XOR as the group operation.

2.3.2 Function Family Ensembles

A function family ensemble is an infinite set of function families, whose elements (families) are typically indexed by the set of integers. Let $\mathcal{F} = \{\mathcal{F}_n\colon \mathcal{D}_n \mapsto \mathcal{R}_n\}_{n\in\mathbb{N}}$ stands for an ensemble of function families, where each $f \in \mathcal{F}_n$ has domain $\mathcal{D}_n$ and its range is a subset of $\mathcal{R}_n$. Such ensemble is *length preserving*, if $\mathcal{D}_n = \mathcal{R}_n = \{0,1\}^n$ for every n. We naturally extend Definitions 2.2 and 2.3 to function family ensembles.

For a function family ensemble to be useful it should have an efficient sampling and evaluation algorithms.

Definition 2.4 (efficient function family ensembles)**.** *A function family ensemble $\mathcal{F} = \{\mathcal{F}_n\colon \mathcal{D}_n \mapsto \mathcal{R}_n\}_{n\in\mathbb{N}}$ is* efficient*, if the following hold:*

Efficient sampling. *$\mathcal{F}$ is samplable in polynomial-time: there exists a* PPTM *that given 1^n, outputs (the description of) a uniform element in $\mathcal{F}_n$.*

Efficient evaluation. *There exists a deterministic algorithm that given $x \in \mathcal{D}_n$ and (a description of) $f \in \mathcal{F}_n$, runs in time* $\mathrm{poly}(n, |x|)$ *and outputs $f(x)$.*

2.3.3 Many-Wise Independent Hashing

Definition 2.5 (k-wise independent families)**.** *A function family $\mathcal{H} = \{h\colon \mathcal{D} \mapsto \mathcal{R}\}$ is* k-wise independent *(with respect to $\mathcal{D}$ and $\mathcal{R}$), if*

$$\Pr_{h\leftarrow\mathcal{H}}[h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge \ldots \wedge h(x_k) = y_k] = \frac{1}{|\mathcal{R}|^k},$$

for every distinct $x_1, x_2, \ldots, x_k \in \mathcal{D}$ and every $y_1, y_2, \ldots, y_k \in \mathcal{R}$.

For every $\ell, k \in \text{poly}$, the existence of efficient $k(n)$-wise independent family ensembles mapping strings of length $\ell(n)$ to strings of length n is well known ([11, 52]). A simple and well known example of k-wise independent functions is the collection of all polynomials of degree $(k-1)$ over a finite field. This construction has small size, and each evaluation of a function at a given point requires k operations in the field.

Fact 2.6. *For $\ell, n, k \in \mathbb{N}$, there exists an k-wise independent function family $\mathcal{H} = \{h\colon \{0,1\}^\ell \mapsto \{0,1\}^n\}$, such that sampling a random element in $\mathcal{H}$ requires $k \cdot \max\{\ell, n\}$ random bits, and evaluating a function from $\mathcal{H}$ is done in time* $\text{poly}(\ell, n, k)$.

We mention that a k-wise independent families (as defined in Definition 2.5) look random for k-query distinguishers, *both* non-adaptive and adaptive ones. On the other hand, *almost* k-wise independent families[6] are only granted to be resistant against *non-adaptive* distinguishers.[7] Yet, the result presented in Section 3 yields that, in some cases, the adaptive security of the latter families follows from their non-adaptive security.

2.4 Pseudorandom Functions

Definition 2.7 (Pseudorandom Functions)**.** *An efficient function family ensemble $\mathcal{F} = \{\mathcal{F}_n\colon \{0,1\}^{m(n)} \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ is an (adaptive) (q,t,ε)-*PRF*, if for every t-time q-query oracle-aided algorithm (distinguisher)* D*, it holds that*

$$\left|\Pr_{f\leftarrow\mathcal{F}_n}[\mathsf{D}^f(1^n) = 1] - \Pr_{\pi\leftarrow\Pi_{m(n),\ell(n)}}[\mathsf{D}^\pi(1^n) = 1]\right| \leq \varepsilon(n),$$

for large enough n. If $q(n)$ is only bounded by $t(n)$ for every $n \in \mathbb{N}$, then $\mathcal{F}$ is called (t,ε)-PRF. A $(t, 1/t)$-PRF is called a t-PRF.

If D *is limited to be* non-adaptive *(i.e., it has to write all his oracle calls before making the first call), then $\mathcal{F}$ is called* non-adaptive (q,t,ε)-PRF *(and we apply the above notational conventions also for this case).*

Some applications require the pseudorandom functions to be secure against distinguisher with access to many oracles (and not just a single oracle as in Definition 2.7).

[6] Formally, a function family $\mathcal{H} = \{h : \mathcal{D} \mapsto \mathcal{R}\}$ is (ε, k)-wise independent if for any $x_1, \ldots, x_k \in \mathcal{D}$ and for any $y_1, \ldots, y_k \in \mathcal{R}$ it holds that $\left|\Pr_{h\leftarrow\mathcal{H}}[h(x_1) = y_1 \wedge \cdots \wedge h(x_k) = y_k] - |\mathcal{R}|^{-k}\right| \leq \varepsilon$. We call a family of functions an almost k-wise independent family, if it is (ε, k)-wise independent for some small $\varepsilon > 0$.

[7] See Footnote 3 and references therein.

Definition 2.8 (pseudorandom functions secure against many-oracle distinguishers)**.** *An efficient function family ensemble* $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{m(n)} \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ *is an* s-oracle (q,t,ε)-PRF*, if for every* t*-time* q*-query* s*-oracle-aided algorithm (distinguisher)* D*, it holds that*

$$\left|\Pr_{\overline{f}\leftarrow\mathcal{F}_n^s}[\mathsf{D}^{\overline{f}}(1^n)=1] - \Pr_{\overline{\pi}\leftarrow\Pi^s_{m(n),\ell(n)}}[\mathsf{D}^{\overline{\pi}}(1^n)=1]\right| \leq \varepsilon(n),$$

for large enough n*.*

The following lemma shows that a standard (single oracle) PRF is also a many-oracle one, with only slight worse parameters.

Lemma 2.9 (cf., [9], Theorem 1)**.** *Let* $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{m(n)} \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ *be a function family ensemble. Then for every* t*-time* q*-query* s-oracle *adaptive [resp., non-adaptive] distinguisher* D*, there exists a* $(t + s\cdot q\cdot e_{\mathcal{F}})$*-time* q*-query* single-oracle *adaptive [resp., non-adaptive] distinguisher* $\widehat{\mathsf{D}}$*, where* $e_{\mathcal{F}}$ *stands for the evaluation time of* $\mathcal{F}$*,*[8] *with*

$$\begin{aligned}&\left|\Pr_{f\leftarrow\mathcal{F}_n}[\widehat{\mathsf{D}}^{f}(1^n)=1] - \Pr_{\pi\leftarrow\Pi_{m(n),\ell(n)}}[\widehat{\mathsf{D}}^{\pi}(1^n)=1]\right| \\ &\geq \frac{1}{s}\cdot\left|\Pr_{\overline{f}\leftarrow\mathcal{F}_n^s}[\mathsf{D}^{\overline{f}}(1^n)=1] - \Pr_{\overline{\pi}\leftarrow\Pi^s_{m(n),\ell(n)}}[\mathsf{D}^{\overline{\pi}}(1^n)=1]\right|,\end{aligned}$$

for every $n\in\mathbb{N}$*.*

Lemma 2.9 is proven using a standard hybrid argument. For a complete proof see [9].[9]

3 From Non-Adaptive to Adaptive Hashing

In this section we show that non-adaptively secure function families with a certain combinatorial property are also adaptively secure, yielding that the function families of Pagh and Pagh [42] and Aumüller et al. [4] are adaptively secure. In the next sections we take advantage of the latter implication to derive our hardness-preserving PRF reductions. We note that the above approach is not useful for (arbitrary) non-adaptive PRFs, since a PRF might no posses the required combinatorial property (indeed, not every non-adaptive PRF is an adaptive one).

To define the aforementioned combinatorial property, we use the notion of left-monotone sets.

Definition 3.1 (left-monotone sets)**.** *Let* $\mathcal{S}$ *and* $\mathcal{T}$ *be sets. A set* $\mathcal{M}\subseteq\mathcal{S}^*\times\mathcal{T}$ *is* left-monotone*, if for every* $(\overline{s_1},t)\in\mathcal{M}$ *and every* $\overline{s_2}\in\mathcal{S}^*$ *that has* $\overline{s_1}$ *as a prefix, it holds that* $(\overline{s_2},t)\in\mathcal{M}$*.*

Namely, a product set is left monotone, if it is monotone with respect to its left-hand-side part, where all sequences having a prefix in a monotone set, are also in the set. The main result of this section is stated as follows.

Lemma 3.2. *Let* $\mathcal{U}$ *and* $\mathcal{V}$ *be non-empty sets, let* $\mathcal{F} = \mathcal{F}(\mathcal{U},\mathcal{V}) = \{f_{u,v}\colon\mathcal{D}\mapsto\mathcal{R}\}_{(u,v)\in\mathcal{U}\times\mathcal{V}}$ *be a function family and let* BAD $\subseteq\mathcal{D}^*\times\mathcal{U}$ *be left monotone. Let* $t\in\mathbb{N}$*, and assume that for every* $\overline{q} = (q_1,\ldots,q_{|\overline{q}|})\in\mathcal{D}^{\leq t}$*:*[10]

[8] That is, $\mathsf{D}(1^n)$ runs in time $t(n)+s\cdot q(n)\cdot e_{\mathcal{F}}(n)$. Moreover, we implicitly assume that the evaluation time of $\mathcal{F}$ is greater than the time needed to sample $\ell(n)$ random bits.

[9] [9] only states, and proves, the adaptive case, but the very same lines also yields the non-adaptive case.

[10] Recall that for a set $\mathcal{S}$ and an integer t, $\mathcal{S}^{\leq t}$ denotes the set $\{\overline{s}\in\mathcal{S}^*\colon\ |\overline{s}|\leq t\ \ \wedge\ \ \overline{s}[i]\neq\overline{s}[j]\ \ \forall i\neq j\in[|\overline{s}|]\}$.

1. $\left(f(q_1),\ldots,f(q_{|\overline{q}|})\right)_{f\leftarrow\{f_{u,v}\colon v\in\mathcal{V}\}}$ *is uniform over* $\mathcal{R}^{|\overline{q}|}$*, for every* $u\in\mathcal{U}$ *with* $(\overline{q},u)\notin \mathrm{BAD}$*, and*
2. $\Pr_{u\leftarrow\mathcal{U}}[(\overline{q},u)\in\mathrm{BAD}]\leq\varepsilon$*.*

Then

$$\left|\Pr_{\substack{u\leftarrow\mathcal{U}\\ v\leftarrow\mathcal{V}}}[\mathsf{D}^{f_{u,v}}=1]-\Pr_{\pi\leftarrow\Pi}[\mathsf{D}^{\pi}=1]\right|\leq\varepsilon,$$

for every t*-query oracle-aided* adaptive *algorithm* D*, letting* Π *be the set of all functions from* $\mathcal{D}$ *to* $\mathcal{R}$*.*

Note that the above properties of the family $\mathcal{F}$ mean that $\mathcal{F}$ is non-adaptively secure. The proof that $\mathcal{F}$ is also *adaptively* secure, which critically uses the above structure of the set BAD, can be found in Appendix A.[11]

3.1 The Pagh and Pagh [42] Function Family

We show that Lemma 3.2 can be applied to the function family of Pagh and Pagh [42].

Definition 3.3 (The Pagh and Pagh [42] function family)**.** *Let* $\mathcal{H}$ *be a function family from* $\mathcal{D}$ *to* $\mathcal{U}$*, let* $\mathcal{G}$ *be a function family from* $\mathcal{D}$ *to* $\mathcal{R}$ *and let* $\mathcal{F}$ *be a function family from* $\mathcal{S}$ *to* $\mathcal{R}$*, with* $\mathcal{U}\subseteq\mathcal{S}$ *and* $\mathcal{R}$ *being a group with respect to the operation* $\oplus$*. The function family* $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ *from* $\mathcal{D}$ *to* $\mathcal{R}$*, is defined by*

$$\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F}):=(\mathcal{F}\circ\mathcal{H})\oplus(\mathcal{F}\circ\mathcal{H})\oplus\mathcal{G}.$$

For $h_1,h_2\in\mathcal{H}$*, let* $\mathcal{PP}_{h_1,h_2}(\mathcal{G},\mathcal{F}):=(\mathcal{F}\circ h_1)\oplus(\mathcal{F}\circ h_2)\oplus\mathcal{G}$*.*

Graphically, this function family is given in Figure 1. Pagh and Pagh [42] showed that when instantiated with the proper function families, the above function family has the following properties:

Theorem 3.4 ([42])**.** *Let* $t\in\mathbb{N}$*, let* $\mathcal{H}=\{h\colon\mathcal{D}\mapsto\mathcal{U}\}$ *and* $\mathcal{G}=\{g\colon\mathcal{D}\mapsto\mathcal{R}\}$ *be function families with* $\mathcal{R}$ *being a group with respect to the operation* $\oplus$*, and let* $\Pi=\Pi_{\mathcal{S}\mapsto\mathcal{R}}$*.*

If $\mathcal{U}\subseteq\mathcal{S}$ *and* $|\mathcal{U}|\geq 4t$*, then for every* $k\in\mathbb{N}$ *there exists a left-monotone set* $\mathrm{BAD}\subseteq\mathcal{D}^{\leq t}\times\mathcal{H}^2$ *such that the following holds for every* $\overline{q}=(q_1,\ldots,q_{|\overline{q}|})\in\mathcal{D}^{\leq t}$*:*

1. *Assuming that* $\mathcal{G}$ *is* k*-wise independent over the elements of* $\overline{q}$*, then* $\left(f(q_1),\ldots,f(q_{|\overline{q}|})\right)_{f\leftarrow\mathcal{PP}_{h_1,h_2}(\mathcal{G},\Pi)}$ *is uniform over* $\mathcal{R}^{|\overline{q}|}$ *for every* $u\in\mathcal{U}$ *such that* $(\overline{q},u)\notin\mathrm{BAD}$*.*
2. *Assuming that* $\mathcal{H}$ *is* k*-wise independent over the elements of* $\overline{q}$*, then* $\Pr_{u\leftarrow\mathcal{H}^2}[(\overline{q},u)\in\mathrm{BAD}]\leq t/2^{\Omega(k)}$*.*[12]

[11] Lemma 3.2 can be derived as a special case of a result given in [26, Theorem 12] (closing a gap in the proof appearing in [32]). Yet, for the sake of completeness, we include an independent proof of this lemma here.

[12] The function family we consider above (i.e., $\mathcal{PP}$) is slightly different than the one given in [42]. Their construction maps element $x\in\mathcal{D}$ to $F_1[h_1(x)]\oplus F_2[h_2(x)]\oplus g(x)$, where F_1 and F_2 are uniformly chosen vectors from $\mathcal{R}^t$, $h_1,h_2\colon\mathcal{D}\mapsto[t]$ are uniformly chosen from a function family $\mathcal{H}$ and $g\colon\mathcal{D}\mapsto\mathcal{R}$ is chosen uniformly from a function family $\mathcal{G}$. Yet, the correctness of Theorem 3.4 follows in a straightforward manner from [42] original proof (specifically from Lemma 3.3 and Lemma 3.4).

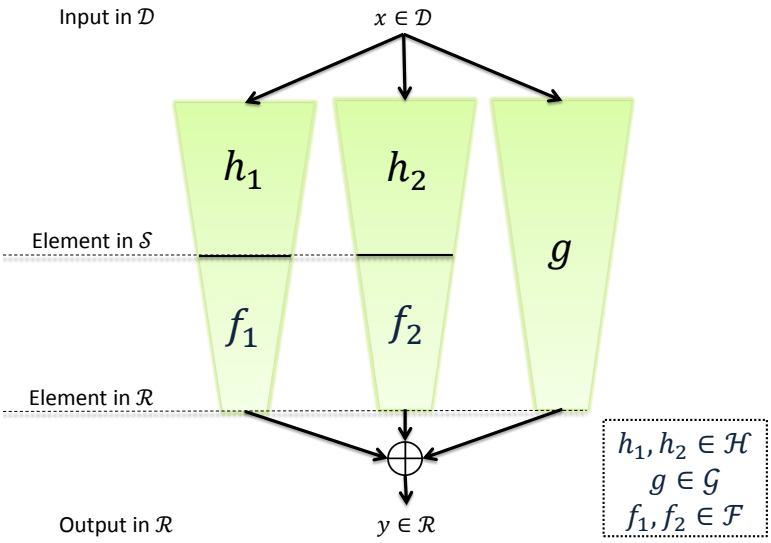

Figure 1: The function family $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F})$. $\mathcal{H}$ hashes down a domain $\mathcal{D}$ to a domain $\mathcal{S}$. Then $\mathcal{F}$ maps $\mathcal{S}$ to $\mathcal{R}$. We do this twice and xor it with $\mathcal{G}$, that hashes the domain $\mathcal{D}$ directly to $\mathcal{R}$.

Pagh and Pagh [42] concluded that for (the many) applications where the analysis is applied with respect to a static set it is safe to use this family instead. However, as we can see, the function family $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \Pi)$ is not only close to being uniform in the eyes of a non-adaptive distinguisher, but also allows us to apply Lemma 3.2 to deduce its security in the eyes of adaptive distinguishers. By plugging in Theorem 3.4 into the general framework lemma (Lemma 3.2), we get the following result:

Lemma 3.5. *Let $t \in \mathbb{N}$, let $\mathcal{H}$, $\mathcal{G}$ and Π be as in Theorem 3.4, and let D be an* adaptive*, t-query oracle-aided algorithm. Assuming that $\mathcal{H}$ and $\mathcal{G}$ are k-wise independent, then*

$$\left|\Pr_{f \leftarrow \mathcal{PP}(\mathcal{H},\mathcal{G},\Pi)}[\mathsf{D}^f = 1] - \Pr_{\pi \leftarrow \Pi}[\mathsf{D}^\pi = 1]\right| \leq t/2^{\Omega(k)}.$$

Proof. Let $\mathcal{U} = \mathcal{H} \times \mathcal{H}$ and $\mathcal{V} = \Pi \times \Pi \times \mathcal{G}$. For $(h_1, h_2) \in \mathcal{U}$ and $(\pi_1, \pi_2, g) \in \mathcal{V}$, let $F_{(h_1,h_2),(\pi_1,\pi_2,g)} = \pi_1 \circ h_1 \oplus \pi_2 \circ h_2 \oplus g$, and let $\mathcal{F} = \{F_{u,v} \colon \mathcal{D} \mapsto \mathcal{R}\}_{(u,v)\in\mathcal{U}\times\mathcal{V}}$. Finally, let BAD be the set BAD of Theorem 3.4. We prove the lemma showing that the above sets meet the requirements stated in Lemma 3.2.

Item 1 of Theorem 3.4 and the assumed independence of $\mathcal{G}$ and $\mathcal{H}$, yield that the first requirement of Lemma 3.2 is satisfied. Item 2 of Theorem 3.4 yields that the second requirement of Lemma 3.2 is satisfied for $\varepsilon = t/2^{\Omega(k)}$. Hence, the proof of the lemma follows by Lemma 3.2. □

Remark 3.6. *For some of our applications, see Sections 4 to 6, we need to apply Lemma 3.5 with efficient k-wise independent function family ensembles mapping strings of length n to the set $[t(n)]_{\{0,1\}^n}$, where t is an efficiently computable function. It is easy to see (cf., [7]) that such ensembles exist for any efficiently computable t that is a power of two. By considering $t'(n) = 2^{\lfloor \log(t(n)) \rfloor}$, we use these ensembles for our applications, while only causing factor of two loss in the resulting security.*

We use the above function family of Pagh and Pagh [42] to extend the domain of pseudorandom functions (see Section 4), and to transform a non-adaptive pseudorandom function into an adaptive one (see Section 5). In Section 6 we instantiate the above framework with the more advanced function family of Aumüller et al. [4], to get more versatile variants of the above applications.

4 PRF Domain Extension

In this section we use the function family $\mathcal{PP}$ of Pagh and Pagh [42] (see Section 3) to extend a domain of a given PRF. We start with instantiation of $\mathcal{PP}$ in a settings of strings in $\{0,1\}^*$.

Corollary 4.1. *Let k, d, s and r be integers, let $\mathcal{H} = \{h\colon \{0,1\}^d \mapsto \{0,1\}^s\}$ and $\mathcal{G} = \{g\colon \{0,1\}^d \mapsto \{0,1\}^r\}$ be function families. Assume that $\mathcal{H}$ and $\mathcal{G}$ are k-wise independent function families, and let $\mathcal{PP}$ be as in Definition 3.3. Then for any q-query adaptive distinguisher D, it holds that*

$$\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi_{s,r})}[\mathsf{D}^f = 1] - \Pr_{\pi\leftarrow\Pi_{d,r}}[\mathsf{D}^\pi = 1]\right| \leq q/2^{\Omega(k)},$$

for any $q \leq 2^{s-2}$.

Proof. The proof follows Lemma 3.5. Set $t = q$, $\mathcal{D} = \{0,1\}^d$, $\mathcal{U} = \mathcal{S} = \{0,1\}^s$ and $\mathcal{R} = \{0,1\}^r$. Note that $|\mathcal{U}| = 2^s \geq 4q = 4t$. Hence, the proof follows a simple implication of Lemma 3.5. □

The above corollary shows that $\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi)$, where Π is the set of all functions with domain $\{0,1\}^s$ and range $\{0,1\}^r$, is statistically close to the set of all functions with domain $\{0,1\}^d$ and range $\{0,1\}^r$, where s can be smaller than d. Our next step is, given a PRF $\mathcal{F}$, to show that $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ is computationally close to $\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi)$. Recall that if $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{F}$ are ensembles of function families indexed by an integer n, then $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ is also an ensemble of function families, and we denoted its n'th function family as $\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)$.

Lemma 4.2. *Let $\mathcal{H} = \{\mathcal{H}_n\colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$, $\mathcal{G} = \{\mathcal{G}_n\colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ and $\mathcal{F} = \{\mathcal{F}_n\colon \{0,1\}^{s(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ be function families. Then, for every t-time q-query oracle-aided distinguisher D, there exists a $(t + 2q\cdot(\mathsf{e}_\mathcal{H} + \mathsf{e}_\mathcal{G} + \mathsf{e}_\mathcal{F}))$-time q-query distinguisher $\widehat{\mathsf{D}}$, where $\mathsf{e}_\mathcal{H}, \mathsf{e}_\mathcal{G}, \mathsf{e}_\mathcal{F}\colon \mathbb{N} \mapsto \mathbb{N}$ are the evaluation and sampling times of $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{F}$ respectively,*[13] *with*

$$\begin{aligned}&\left|\Pr_{f\leftarrow\mathcal{F}_n}[\widehat{\mathsf{D}}^f(1^n) = 1] - \Pr_{\pi\leftarrow\Pi_{s(n),r(n)}}[\widehat{\mathsf{D}}^\pi(1^n) = 1]\right| \\ &\geq \frac{1}{2}\cdot\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n) = 1] - \Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\Pi_{s(n),r(n)})}[\mathsf{D}^f(1^n) = 1]\right|,\end{aligned}$$

for every $n \in \mathbb{N}$.

Proof. Let $\widetilde{\mathsf{D}}$ be the following two-oracle distinguisher:

Algorithm 4.3 ($\widetilde{\mathsf{D}}$)**.**

Input: 1^n.

Oracle: *functions ϕ_1, ϕ_2 from $\mathcal{D}_n$ to $\mathcal{R}_n$.*

[13] That is, e.g., $\mathsf{e}_\mathcal{H}(n)$ is an upper bound for the time it takes to sample $h \leftarrow \mathcal{H}_n$, as well as the time it takes to compute $h(x)$ for every $h \in \mathcal{H}_n$ and $x \in \{0,1\}^{d(n)}$.

1. *Set* $h_1, h_2 \leftarrow \mathcal{H}_n$, $g \leftarrow \mathcal{G}_n$.
2. *Set* $f = (\phi_1 \circ h_1) \oplus (\phi_2 \circ h_2) \oplus g$.
3. *Emulate* $\mathsf{D}^f(1^n)$.

Note that $\widetilde{\mathsf{D}}(1^n)$ makes $q(n)$ queries to ϕ_1 and ϕ_2, and it can be implemented to run in time $t(n) + q(n) \cdot (\mathsf{e}_{\mathcal{H}}(n) + \mathsf{e}_{\mathcal{G}}(n))$. Observe that in case ϕ_1 and ϕ_2 are uniformly drawn from $\mathcal{F}_n$, then the emulation of $\mathsf{D}^f(1^n)$ done in $\widetilde{\mathsf{D}}^{\phi_1,\phi_2}(1^n)$ is identical to a random execution of $\mathsf{D}^f(1^n)$ with $f \leftarrow \mathcal{PP}(\mathcal{H}_n, \mathcal{G}_n, \mathcal{F}_n)$. Similarly, in case ϕ_1 and ϕ_2 are uniformly drawn from $\Pi_{s(n),r(n)}$, then the emulation is identical to a random execution of $\mathsf{D}^f(1^n)$ with $f \leftarrow \mathcal{PP}(\mathcal{H}_n, \mathcal{G}_n, \Pi_{s(n),r(n)})$. Thus,

$$\begin{aligned}&\left|\Pr_{(f_1,f_2)\leftarrow\mathcal{F}_n\times\mathcal{F}_n}[\widetilde{\mathsf{D}}^{f_1,f_2}(1^n)=1] - \Pr_{(\pi_1,\pi_2)\leftarrow\Pi_{s(n),r(n)}\times\Pi_{s(n),r(n)}}[\widetilde{\mathsf{D}}^{\pi_1,\pi_2}(1^n)=1]\right| \\ =&\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\Pi_{s(n),r(n)})}[\mathsf{D}^f(1^n)=1]\right|.\end{aligned} \tag{2}$$

Hence, Lemma 2.9 yields that there exists a single-oracle distinguisher $\widehat{\mathsf{D}}$ that when invoked on input of length n makes $q(n)$ queries to its oracle and runs in time $t(n) + q(n) \cdot (\mathsf{e}_{\mathcal{H}}(n) + \mathsf{e}_{\mathcal{G}}(n)) + 2q(n) \cdot \mathsf{e}_{\mathcal{F}}(n) \leq t(n) + 2q(n) \cdot (\mathsf{e}_{\mathcal{H}}(n) + \mathsf{e}_{\mathcal{G}}(n) + \mathsf{e}_{\mathcal{F}}(n))$, such that

$$\begin{aligned}&\left|\Pr_{f\leftarrow\mathcal{F}_n}[\widehat{\mathsf{D}}^f(1^n)=1] - \Pr_{\pi\leftarrow\Pi_{s(n),r(n)}}[\widehat{\mathsf{D}}^{\pi}(1^n)=1]\right| \\ \geq&\frac{1}{2}\cdot\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\Pi_{s(n),r(n)})}[\mathsf{D}^f(1^n)=1]\right|.\end{aligned}$$

□

Assuming that $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{s(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ is (q,t,ε)-PRF, Lemma 4.2 yields that for every $(t + 2q(\mathsf{e}_{\mathcal{H}} + \mathsf{e}_{\mathcal{G}} + \mathsf{e}_{\mathcal{F}}))$-time distinguisher D, it holds that

$$\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\Pi_{s(n),r(n)})}[\mathsf{D}^f(1^n)=1]\right| \leq 2\varepsilon(n),$$

for large enough n. Hence, assuming $\mathcal{H}$ and $\mathcal{G}$ are k-wise independent function families, Corollary 4.1 and the triangle inequality yield that

$$\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{\pi\leftarrow\Pi_{d(n),r(n)}}[\mathsf{D}^{\pi}(1^n)=1]\right| \leq 2\varepsilon(n) + q(n)/2^{\Omega(k(n))}$$

for large enough n. We get the following domain extension using the $\mathcal{PP}$ construction.

Theorem 4.4 (Restating Theorem 1.1)**.** *Let* $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ *and* $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ *be efficient* $k(n)$*-wise independent function family ensembles, and let* $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{s(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ *be a* (q,t,ε)*-PRF. Then,*

$$\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F}) = \{\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n) \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$$

is a $(q, t - p \cdot q, 2\varepsilon + q/2^{\Omega(k)})$*-PRF, where* p *is a polynomial determined by the evaluation and sampling time of* $\mathcal{H}$*,* $\mathcal{G}$ *and* $\mathcal{F}$ *and* $q(n) \leq 2^{s(n)-2}$ *for every* $n \in \mathbb{N}$*.*

Note that in order for Theorem 4.4 to be useful, we have to set $k(n) = \Omega(\log q(n))$. In Section 6 we show how to achieve domain extension using functions with less independence, but with the cost of additional calls to the PRF.

5 From Non-Adaptive to Adaptive PRF

In this section we use the function family $\mathcal{PP}$ of Pagh and Pagh [42] (see Section 3), to transform an non-adaptive PRF into an adaptive one in a security preserving manner. As in Section 4, we start with instantiating $\mathcal{PP}$ in a convenient setting. To ease notations, we assume that the given non-adaptive PRF is length preserving.

Corollary 5.1. *Let k, q, d and n be integers with $q \leq 2^{n-2}$, let $\mathcal{H} = \{h\colon \{0,1\}^d \mapsto [4q]_{\{0,1\}^n}\}$ and let $\mathcal{G} = \{g\colon \{0,1\}^d \mapsto \{0,1\}^n\}$. Assume that $\mathcal{H}$ and $\mathcal{G}$ are k-wise independent function families, then for any q-query adaptive distinguisher D, it holds that*

$$\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi_n)}[\mathsf{D}^f = 1] - \Pr_{\pi\leftarrow\Pi_{d,n}}[\mathsf{D}^\pi = 1]\right| \leq q/2^{\Omega(k)}.$$

Proof. The proof follows from Lemma 3.5. Set $t = q$, $\mathcal{D} = \{0,1\}^d$, $\mathcal{S} = \{0,1\}^n$, $\mathcal{U} = [4q]_{\{0,1\}^n}$ and $\mathcal{R} = \{0,1\}^n$. Note that $|\mathcal{U}| = 4q = 4t$. □

We begin by showing that $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ is computationally indistinguishable from $\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi)$, where $\mathcal{F}$ is *non-adaptive* pseudorandom function and Π is a truly random function with the same domain and range as $\mathcal{F}$. Recall that if $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{F}$ are ensembles of function families indexed by an integer n, then $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ is also an ensemble of function families, and we denoted its nth function family as $\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)$.

Lemma 5.2. *Let q and d be integer functions, let $\mathcal{H} = \{\mathcal{H}_n\colon \{0,1\}^{d(n)} \mapsto [4q(n)]_{\{0,1\}^n}\}_{n\in\mathbb{N}}$, $\mathcal{G} = \{\mathcal{G}_n\colon \{0,1\}^{d(n)} \mapsto \{0,1\}^n\}$, and $\mathcal{F} = \{\mathcal{F}_n\colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be function families. Then for every t-time q-query oracle-aided adaptive distinguisher D, there exists a $(\mathsf{e}_q + t + 8q(\mathsf{e}_\mathcal{H} + \mathsf{e}_\mathcal{G} + \mathsf{e}_\mathcal{F}))$-time $4q$-query, non-adaptive, oracle-aided distinguisher $\widehat{\mathsf{D}}$, where e_q is the evaluation time of q, and $\mathsf{e}_\mathcal{H}$, $\mathsf{e}_\mathcal{G}$ and $\mathsf{e}_\mathcal{F}$ are the sampling and evaluation time of $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{F}$ respectively, with*

$$\begin{aligned}&\left|\Pr_{f\leftarrow\mathcal{F}_n}[\widehat{\mathsf{D}}^f(1^n) = 1] - \Pr_{\pi\leftarrow\Pi_n}[\widehat{\mathsf{D}}^\pi(1^n) = 1]\right| \\ &\geq \frac{1}{2}\cdot\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n) = 1] - \Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\Pi_n)}[\mathsf{D}^f(1^n) = 1]\right|,\end{aligned}$$

for every $n \in \mathbb{N}$ and $q(n) \leq 2^{n-2}$.

Proof. The proof follows along similar lines to the proof of [7, Lemma 3.3]. Let $\widetilde{\mathsf{D}}$ be the following two-oracle distinguisher:

Algorithm 5.3 ($\widetilde{\mathsf{D}}$)**.**

Input: 1^n.

Oracles: *Functions ϕ_1 and ϕ_2 from $\{0,1\}^n$ to $\{0,1\}^n$.*

1. *Compute $\phi_1(x)$ and $\phi_2(x)$ for every $x \in [4q(n)]_{\{0,1\}^n}$.*
2. *Set $f = (\phi_1 \circ h_1) \oplus (\phi_2 \circ h_2) \oplus g$, where $h \leftarrow \mathcal{H}_n$ and $g \leftarrow \mathcal{G}_n$.*
3. *Emulate $\mathsf{D}^f(1^n)$: answer a query x to ϕ_1 and ϕ_2 made by D with $f(x)$, using the information obtained in Step 1.*

Note that $\widetilde{\mathsf{D}}(1^n)$ makes $4q(n)$ *non-adaptive* queries to ϕ_1 and ϕ_2, and it can be implemented to run in time $\mathsf{e}_q(n) + 8q(n) + t(n) + q(n) \cdot (\mathsf{e}_{\mathcal{H}}(n) + \mathsf{e}_{\mathcal{G}}(n))$. Observe that the definition of $\mathcal{PP}$ guarantees that ϕ_1 and ϕ_2 are only queried on the first $4q(n)$ elements of their domain. Hence, in case ϕ_1 and ϕ_2 are uniformly drawn from $\mathcal{F}_n$, then the emulation of $\mathsf{D}^f(1^n)$ done in $\widetilde{\mathsf{D}}^{\phi_1,\phi_2}$ is identical to a random execution of $\mathsf{D}^f(1^n)$ with $f \leftarrow \mathcal{PP}(\mathcal{H}_n, \mathcal{G}_n, \mathcal{F}_n)$. Similarly, in case ϕ_1 and ϕ_2 are uniformly drawn from Π_n, then the emulation is identical to a random execution of $\mathsf{D}^f(1^n)$ with $f \leftarrow \mathcal{P}(\mathcal{H}_n, \mathcal{G}_n, \Pi_n)$. Thus,

$$\begin{aligned}&\left|\Pr_{(f_1,f_2)\leftarrow\mathcal{F}_n\times\mathcal{F}_n}[\widetilde{\mathsf{D}}^{f_1,f_2}(1^n)=1] - \Pr_{(\pi_1,\pi_2)\leftarrow\Pi_n\times\Pi_n}[\widetilde{\mathsf{D}}^{\pi_1,\pi_2}(1^n)=1]\right| \\ &= \left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{f\leftarrow\mathcal{P}(\mathcal{H}_n,\mathcal{G}_n,\Pi_n)}[\mathsf{D}^f(1^n)=1]\right|.\end{aligned} \tag{3}$$

Hence, Lemma 2.9 yields that there exists a non-adaptive, single-oracle distinguisher $\widehat{\mathsf{D}}$ that when invoked on input of length n makes $4q(n)$ queries and runs in time $\mathsf{e}_q(n) + 8q(n) + t(n) + q(n) \cdot (\mathsf{e}_{\mathcal{H}}(n) + \mathsf{e}_{\mathcal{G}}(n)) + 2q(n) \cdot \mathsf{e}_{\mathcal{F}}(n) \leq \mathsf{e}_q(n) + t(n) + 8q(n) \cdot (\mathsf{e}_{\mathcal{H}}(n) + \mathsf{e}_{\mathcal{G}}(n) + \mathsf{e}_{\mathcal{F}}(n))$, such that

$$\begin{aligned}&\left|\Pr_{f\leftarrow\mathcal{F}_n}[\widehat{\mathsf{D}}^f(1^n)=1] - \Pr_{\pi\leftarrow\Pi_n}[\widehat{\mathsf{D}}^{\pi}(1^n)=1]\right| \\ &\geq \frac{1}{2}\cdot\left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{f\leftarrow\mathcal{P}(\mathcal{H}_n,\mathcal{G}_n,\Pi_n)}[\mathsf{D}^f(1^n)=1]\right|,\end{aligned}$$

for every $n \in \mathbb{N}$. □

Let q be an integer function with $q(n) \leq 2^{n-2}$ for every $n \in \mathbb{N}$, let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be a non-adaptive $(4q, p \cdot t, \varepsilon)$-PRF, for $p(n) \geq \mathsf{e}_q(n) + t(n) + 8q(n) \cdot (\mathsf{e}_{\mathcal{H}}(n) + \mathsf{e}_{\mathcal{G}}(n) + \mathsf{e}_{\mathcal{F}}(n))$. Lemma 5.2 yields that for every t-time q-query oracle-aided algorithm D, it holds that

$$\left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{f\leftarrow\mathcal{P}(\mathcal{H}_n,\mathcal{G}_n,\Pi_n)}[\mathsf{D}^f(1^n)=1]\right| \leq 2\varepsilon(n),$$

for large enough n. Hence, assuming $\mathcal{H}$ and $\mathcal{G}$ are $k(n)$-wise independent, Corollary 5.1 and the triangle inequality yield that

$$\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{G}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{\pi\leftarrow\Pi_n}[\mathsf{D}^{\pi}(1^n)=1]\right| \leq 2\varepsilon(n) + q(n)/2^{\Omega(k(n))}$$

for large enough $n \in \mathbb{N}$. Setting $k(n) = \Theta(\log q(n))$ yields the following theorem.

Theorem 5.4. *Let q be a polynomial-time computable integer function with $q(n) \leq 2^{n-2}$ for every $n \in \mathbb{N}$, let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto [4q(n)]_{\{0,1\}^n}\}_{n\in\mathbb{N}}$ and $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be efficient $(c \cdot \log q)$-wise independent function family ensembles, where $c > 0$ is universal, and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be a non-adaptive $(4q, p \cdot t, \varepsilon)$-PRF, where $p \in$ poly is determined by the evaluation time of $q, \mathcal{H}, \mathcal{G}$ and $\mathcal{F}$. Then $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F})$ is an adaptive $(q, t, 2\varepsilon + 1/q)$-PRF.*

Theorem 5.4 implies the following simpler corollary.

Corollary 5.5 (Restatement of Theorem 1.2)**.** *Let t be a polynomial-time computable integer function with $t(n) \leq 2^{n-2}$ for every $n \in \mathbb{N}$, let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto [4t(n)]_{\{0,1\}^n}\}_{n\in\mathbb{N}}$ and $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be efficient $(c \cdot \log t)$-wise independent function family ensembles, where $c > 0$ is universal, and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be a non-adaptive $(p \cdot t)$-PRF, where $p \in$ poly is determined by the evaluation time of $t, \mathcal{H}, \mathcal{G}$ and $\mathcal{F}$. Then $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F})$ is an adaptive t-PRF.*

Proof. By definition, $\mathcal{F}$ is also $(4t, p \cdot t, 1/(p \cdot t))$-non-adaptive PRF (we assume that $p(n) \geq 4$ for every $n \in \mathbb{N}$). The proof is now a direct implication of Theorem 5.4. □

6 Hardness-Preserving Reductions via Advanced Cuckoo Hashing

In this section we apply the reductions given in Sections 4 and 5 using the function family of Pagh and Pagh [42], with the function family of Aumüller et al. [4], to get a more versatile reduction (for comparison see Section 6.2.1). Roughly speaking, the function family of Aumüller et al. [4] requires less combinatorial work (i.e., smaller independence) than the Pagh and Pagh [42] family. On the other hand, the function family of Aumüller et al. [4] requires more "randomness" (i.e., has a longer description) and is harder to describe. In Section 6.1 we formally define the hash function family of Aumüller et al. [4], state their (non-adaptive) result, and apply Lemma 3.2 to get an adaptive variant of this result. In Section 6.2 we use the function family of Aumüller et al. [4] to obtain a PRF domain extension, where in Section 6.3 we use this family to get a non-adaptive to adaptive transformation of PRFs.

6.1 The Aumüller et al. [4] Function Family

The function family of Aumüller et al. [4] (building upon Dietzfelbinger and Woelfel [15]) follows the same basic outline as the Pagh and Pagh [42] function family, but uses more complex hash functions. Recall that the members of the Pagh and Pagh [42] function family $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ are of the form $(f_1 \circ h_1) \oplus (f_2 \circ h_2) \oplus g$, for $f_1, f_2 \in \mathcal{F}$, $h_1, h_1, \in \mathcal{H}$ and $g \in \mathcal{G}$. In the function family $\mathcal{ADW}(\mathcal{H},\mathcal{L},\mathcal{G},\mathcal{F},\mathcal{M},\mathcal{Y})$ described below, the role of $h_1, h_2 \in \mathcal{H}$ is taken by some variant of *tabulation hashing* (and not simply from a relatively high k-wise independent family as in [42]). Roughly, at the heart of these functions lies a function $a_{h,\overline{g},\overline{m}} : \mathcal{D} \mapsto \mathcal{S}$ of the form:

$$a_{h,\overline{g},\overline{m}}(x) := h(x) \bigoplus_{1 \le i \le z} m_i(g_i(x)),$$

for $h\colon \mathcal{D} \mapsto \mathcal{S}$, $\overline{g} = (g_1, \cdots, g_z)$ where $g_i\colon \mathcal{D} \mapsto \mathcal{U}$ and $\overline{m} = (m_1, \cdots, m_z)$ where $m_i\colon \mathcal{U} \mapsto \mathcal{S}$. Jumping ahead, the m_i's will be chosen to be random functions (or pseudorandom functions) and the g_i's and h will be chosen from a relatively low independence family. The Aumüller et al. [4] construction uses several functions of the above form that, unlike [42], are chosen in a *correlated* manner (in particular, sharing the *same* function vector $\overline{g}$). The precise definition is:

Definition 6.1 (The Aumüller et al. [4] function family)**.** *For $z \in \mathbb{N}$ let*

1. *$\mathcal{D}$,$\mathcal{U}$ be sets and $\mathcal{S},\mathcal{R}$ be commutative groups defined with respect to an operation $\oplus_{\mathcal{S}}$ and $\oplus_{\mathcal{R}}$, respectively (we will omit the subscript when it is clear);*
2. *function families $\mathcal{H} = \{h\colon \mathcal{D} \mapsto \mathcal{S}\}$, $\mathcal{L} = \{\ell\colon \mathcal{D} \mapsto \mathcal{R}\}$, $\mathcal{F} = \{f\colon \mathcal{S} \mapsto \mathcal{R}\}$, $\mathcal{G} = \{g\colon \mathcal{D} \mapsto \mathcal{U}\}$, $\mathcal{M} = \{m\colon \mathcal{U} \mapsto \mathcal{S}\}$ and $\mathcal{Y} = \{y\colon \mathcal{U} \mapsto \mathcal{R}\}$;*
3. *functions $h_1, h_2 \in \mathcal{H}$, $\ell \in \mathcal{L}$, $f_1, f_2 \in \mathcal{F}$;*
4. *function vector $\overline{g} = (g_1, \cdots, g_z)$, where $g_i \in \mathcal{G}$ for every $1 \le i \le z$;*
5. *function vectors $\overline{m}^1 = (m_1^1, \ldots, m_z^1), \overline{m}^2 = (m_1^2, \ldots, m_z^2)$, where $m_i^j \in \mathcal{M}$ for each $j \in \{1,2\}$ and $1 \le i \le z$; and*
6. *function vector $\overline{y} = (y_1, \ldots, y_z)$, where $y_i \in \mathcal{Y}$ for $1 \le i \le z$.*

Define $\mathsf{adw}_{\overline{m}^1,\overline{m}^2,\overline{y},h_1,h_2,\ell,\overline{g},f_1,f_2} : \mathcal{D} \mapsto \mathcal{R}$ *by*

$$\mathsf{adw}_{\overline{m}^1,\overline{m}^2,y,h_1,h_2,\ell,\overline{g},f_1,f_2} := (f_1 \circ a_{h_1,\overline{g},\overline{m}^1}) \oplus_{\mathcal{R}} (f_2 \circ a_{h_2,\overline{g},\overline{m}^2}) \oplus_{\mathcal{R}} a_{\ell,\overline{g},\overline{y}}, \tag{4}$$

where $a_{h,\overline{g},\overline{m}}(x) := h(x) \bigoplus_{1 \le i \le z} m_i(g_i(x))$.[14]

For $\overline{m}^1, \overline{m}^2 \in \mathcal{M}^z$, $h_1, h_2 \in \mathcal{H}$ *and* $\overline{g} \in \mathcal{G}^z$, *function family* $\mathcal{L}, \mathcal{F}$ *and* $\mathcal{Y}$ *as above, let*

$$\mathcal{ADW}_{z,(\overline{m}^1,\overline{m}^2,h_1,h_2,\overline{g})}(\mathcal{L},\mathcal{F},\mathcal{Y}) := \{\mathsf{adw}_{\overline{m}^1,\overline{m}^2,\overline{y},h_1,h_2,\ell,\overline{g},f_1,f_2} : \overline{y} \in \mathcal{Y}^z, \ell \in \mathcal{L}, f_1, f_2 \in \mathcal{F}\}.$$

Finally, let

$$\begin{aligned}\mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\mathcal{F},\mathcal{M},\mathcal{Y}) := \{&\mathsf{adw}_{\overline{m}^1,\overline{m}^2,\overline{y},h_1,h_2,\ell,\overline{g},f_1,f_2} : \\ &\overline{m}^1, \overline{m}^2 \in \mathcal{M}^z, \overline{y} \in \mathcal{Y}^z, h_1, h_2 \in \mathcal{H}, \\ &\ell \in \mathcal{L}, \overline{g} \in \mathcal{G}^z, f_1, f_2 \in \mathcal{F}\}.\end{aligned}$$

Graphically, this function family is described in Figure 2.

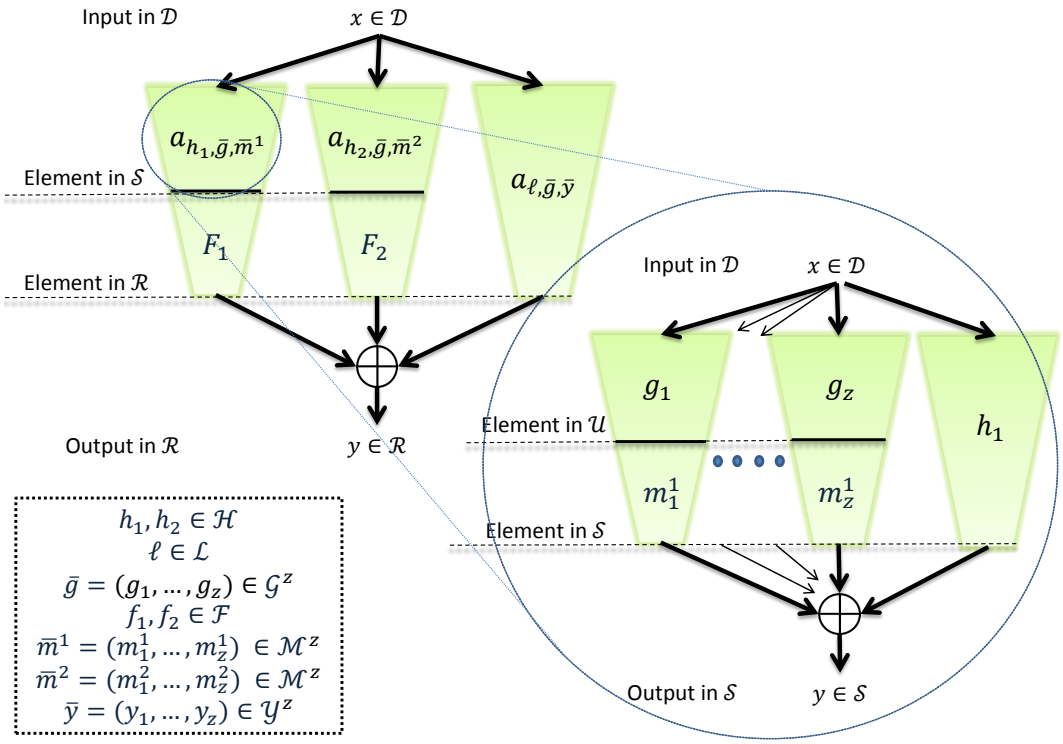

Figure 2: The function family $\mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\mathcal{F},\mathcal{M},\mathcal{Y})$ in the top left corner. On the bottom right corner the function $a_{h_1,\overline{g},\overline{m}^1}$ is depicted. The function $a_{h_2,\overline{g},\overline{m}^2}$ is similar. In $a_{\ell,\overline{g},\overline{y}}$ the range of the functions ℓ and $y_1 \ldots, y_z$ is $\mathcal{R}$ (rather than $\mathcal{S}$).

Aumüller et al. [4] proved the following result with respect to the above function family.

[14] Note that $a_{h_1,\overline{g},\overline{m}^1}, a_{h_2,\overline{g},\overline{m}^2} : \mathcal{D} \mapsto \mathcal{S}$ uses $\oplus_{\mathcal{S}}$ and $a_{\ell,\overline{g},\overline{y}} : \mathcal{D} \mapsto \mathcal{R}$ uses $\oplus_{\mathcal{R}}$.

Theorem 6.2 ([4]). *Let $t, k, z \in \mathbb{N}$ and let $\mathcal{D}, \mathcal{R}, \mathcal{S}, \mathcal{U}$ be commutative groups defined with respect to an operation $\oplus$ such that $|\mathcal{U}| \in [t]$ and $|\mathcal{S}| \geq 4t$.*[15] *Assume that $z \cdot k \in O(\log t)$. Let $\mathcal{H} = \{h \colon \mathcal{D} \mapsto \mathcal{S}\}$, $\mathcal{L} = \{\ell \colon \mathcal{D} \mapsto \mathcal{R}\}$ and $\mathcal{G} = \{g \colon \mathcal{D} \mapsto \mathcal{U}\}$ be $2k$-wise independent hash families. Then there exist a universal constant $\mathsf{const} > 0$ and a left-monotone set $\mathrm{BAD} \subseteq \mathcal{D}^{\leq t} \times \left(((\Pi_{\mathcal{U} \mapsto \mathcal{S}})^z)^2 \times \mathcal{H}^2 \times \mathcal{G}^z\right)$, such that the following holds for every $\overline{q} = (q_1, \ldots, q_{|\overline{q}|}) \in \mathcal{D}^{\leq t}$:*

1. $\left(f(q_1), \ldots, f(q_{|\overline{q}|})\right)_{f \leftarrow \mathcal{ADW}_{z,u}(\mathcal{L}, \Pi_{\mathcal{S} \mapsto \mathcal{R}}, \Pi_{\mathcal{U} \mapsto \mathcal{R}})}$ *is uniform over $\mathcal{R}^{|\overline{q}|}$ for every $u \in ((\Pi_{\mathcal{U} \mapsto \mathcal{S}})^z)^2 \times \mathcal{H}^2 \times \mathcal{G}^z$ such that $(\overline{q}, u) \notin \mathrm{BAD}$, and*
2. $\Pr_{u \leftarrow ((\Pi_{\mathcal{U} \mapsto \mathcal{S}})^z)^2 \times \mathcal{H}^2 \times \mathcal{G}^z}[(\overline{q}, u) \in \mathrm{BAD}] = \mathsf{const} \cdot t / |\mathcal{U}|^{z \cdot k/2}$.

Remark 6.3. *The construction from Definition 6.1 and Theorem 6.2 are taken from [3, Section 6] which is the conference version of [4]. In [3] the authors only considered the case where $z \cdot k$ is constant, independent of t. The proof for the case where $z \cdot k \in O(\log t)$ follows from the proof of [4, Theorem 2].*

Applying Lemma 3.2, the above yields that, for the right choice of parameters, the function family $\mathcal{ADW}_z(\mathcal{H}, \mathcal{L}, \mathcal{F}, \mathcal{G}, \mathcal{M}, \mathcal{Y})$ is not only close to being uniform in the eyes of a *non-adaptive* distinguisher, but also in the eyes of an *adaptive* one. Specifically, combining Lemma 3.2 and theorem 6.2 yields the following result.

Lemma 6.4. *Let $t, k, z, \mathsf{const}, \mathcal{D}, \mathcal{R}, \mathcal{S}, \mathcal{U}, \mathcal{H}, \mathcal{L}, \mathcal{G}$ be as in Theorem 6.2 and let D be an $\mathsf{adaptive}$, t-query oracle-aided algorithm. Then*

$$\left|\Pr_{f \leftarrow \mathcal{ADW}_z(\mathcal{H}, \mathcal{L}, \mathcal{G}, \Pi_{\mathcal{S} \mapsto \mathcal{R}}, \Pi_{\mathcal{U} \mapsto \mathcal{S}}, \Pi_{\mathcal{U} \to \mathcal{R}})}[\mathsf{D}^f = 1] - \Pr_{\pi \leftarrow \Pi}[\mathsf{D}^\pi = 1]\right| = \mathsf{const} \cdot t / |\mathcal{U}|^{z \cdot k/2}.$$

Proof. Let $\mathcal{U}' = ((\Pi_{\mathcal{U} \mapsto \mathcal{S}})^z)^2 \times \mathcal{H}^2 \times \mathcal{G}^z$, $\mathcal{V} = (\Pi_{\mathcal{U} \to \mathcal{R}})^z \times (\Pi_{\mathcal{S} \to \mathcal{R}})^2 \times \mathcal{L}$. For $(\overline{m}^1, \overline{m}^2, h_1, h_2, \overline{g}) \in \mathcal{U}'$ and $(\overline{y}, \pi_1, \pi_2, \ell) \in \mathcal{V}$, let

$$F_{(\overline{m}^1, \overline{m}^2, h_1, h_2, \overline{g}), (\overline{y}, \pi_1, \pi_2, \ell)} = (\pi_1 \circ a_{h_1, \overline{g}, \overline{m}^1}) \oplus (\pi_2 \circ a_{h_2, \overline{g}, \overline{m}^2}) \oplus a_{\ell, \overline{g}, \overline{y}},$$

and let

$$\mathcal{F} = \{F_{u', v} \colon \mathcal{D} \mapsto \mathcal{R}\}_{(u', v) \in \mathcal{U}' \times \mathcal{V}}.$$

Finally, let BAD be the set BAD of Theorem 6.2.

The above sets meet the requirements stated in Lemma 3.2: Item 1 of Theorem 6.2 assures that the first property of Lemma 3.2 is satisfied, and according to Item 2 of Theorem 6.2 we set ε of Lemma 3.2 to be $\mathsf{const} \cdot t / |\mathcal{U}|^{k \cdot z}$, and thus the second property is also satisfied. Hence, applying Lemma 3.2 concludes the proof of the lemma. □

We note that for large enough z, and in contrast to the Pagh and Pagh [42] family, using $\mathcal{ADW}$ we get meaningful results even when using an underlying $k = o(\log t)$-wise independent family. (For a thorough comparison between $\mathcal{PP}$ and $\mathcal{ADW}$ see Section 6.2.1). In particular, for specific settings of parameters we get the following corollaries.

[15] The actual setting in [4] is more general. Specifically, an additional parameter $\varepsilon > 0$ is used to set the size of $\mathcal{S}$ as $(1 + \varepsilon)t$. For simplicity of presentation, comparison with the statement of Theorem 3.4, and since we use this theorem only when m is an integer, we set $\varepsilon = 3$.

Corollary 6.5. *Let $t, \mathsf{const}, \mathcal{D}, \mathcal{R}, \mathcal{S}, \mathcal{U}, \mathcal{H}, \mathcal{L}, \mathcal{G}$ be as in Theorem 6.2. Let $c \in \mathbb{N}$, $z = 2(c+2)$, $|\mathcal{U}| = t$ and $k = 1$. Let D be an* adaptive*, t-query oracle-aided algorithm. Then*

$$\left|\Pr_{f \leftarrow \mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\Pi_{\mathcal{S}\mapsto\mathcal{R}},\Pi_{\mathcal{U}\mapsto\mathcal{S}},\Pi_{\mathcal{U}\mapsto\mathcal{R}})}[\mathsf{D}^f = 1] - \Pr_{\pi\leftarrow\Pi}[\mathsf{D}^\pi = 1]\right| = \mathsf{const}/t^{c+1}.$$

Proof. Since $\mathcal{H}$ and $\mathcal{L}$ are pairwise independent, $z = 2(c+2)$, and $k = 1$ it holds that

$$\mathsf{const} \cdot t / |\mathcal{U}|^{z\cdot k/2} \leq \mathsf{const}/t^{c+1}.$$

Plugging this into Lemma 6.4 completes the proof. □

Corollary 6.6. *Let $t, \mathsf{const}, \mathcal{D}, \mathcal{R}, \mathcal{S}, \mathcal{U}, \mathcal{H}, \mathcal{L}, \mathcal{G}$ be as in Theorem 6.2. Let $c \in \mathbb{N}$, $z = 2(c+2)\cdot\log t$, $k = 1$ and $|\mathcal{U}| = 2$. Let D be an* adaptive*, t-query oracle-aided algorithm. Then*

$$\left|\Pr_{f \leftarrow \mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\Pi_{\mathcal{S}\mapsto\mathcal{R}},\Pi_{\mathcal{U}\mapsto\mathcal{S}},\Pi_{\mathcal{U}\mapsto\mathcal{R}})}[\mathsf{D}^f = 1] - \Pr_{\pi\leftarrow\Pi}[\mathsf{D}^\pi = 1]\right| = \mathsf{const}/t^{c+1}.$$

Proof. Since $\mathcal{H}$ and $\mathcal{L}$ are pairwise independent, $z = 2(c+2)\cdot\log t$, and $k = 1$ it holds that

$$\mathsf{const} \cdot t / |\mathcal{U}|^{z\cdot k/2} \leq \mathsf{const}/t^{c+1}.$$

Plugging this into Lemma 6.4 completes the proof. □

6.2 PRF Domain Extending Via the $\mathcal{ADW}$ Family

In this subsection we present a PRF a domain extension using the Aumüller et al. [4] family, $\mathcal{ADW}$. This allows us to avoid the large independence required for using the Pagh and Pagh [42] family $\mathcal{PP}$.

Instantiating the above construction in the setting of strings over $\{0,1\}^*$, we get the following corollaries, analogous to Corollary 4.1.

Corollary 6.7. *Let $d, u, s, r, q, c > 0$ be integers such that $u \leq s \leq r$, let $z = 2(c+2)$ and let $\mathcal{H} = \{\mathcal{H}\colon \{0,1\}^d \mapsto \{0,1\}^s\}$, $\mathcal{L} = \{\mathcal{L}\colon \{0,1\}^d \mapsto \{0,1\}^r\}$ and $\mathcal{G} = \{\mathcal{G}\colon \{0,1\}^d \mapsto \{0,1\}^u\}$. Assume $\mathcal{H}$,$\mathcal{L}$ and $\mathcal{G}$ are pairwise independent function family, then for any q-query adaptive distinguisher D, it holds that*

$$\left|\Pr_{f \leftarrow \mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\Pi_{s,r},\Pi_{u,s},\Pi_{u,r})}[\mathsf{D}^f = 1] - \Pr_{\pi\leftarrow\Pi_{d,r}}[\mathsf{D}^\pi = 1]\right| \leq O\left(1/q^{c+1}\right),$$

for any $q \leq 2^{r-2}$.

Corollary 6.8. *Let $d, s, r, q, c > 0$ be integers, let $u = 2$, let $z = 2(c+2)\cdot\log q$ and let $\mathcal{H} = \{\mathcal{H}\colon \{0,1\}^d \mapsto \{0,1\}^s\}$, $\mathcal{L} = \{\mathcal{L}\colon \{0,1\}^d \mapsto \{0,1\}^r\}$ and $\mathcal{G} = \{\mathcal{G}\colon \{0,1\}^d \mapsto \{0,1\}^u\}$. Assume $\mathcal{H}$,$\mathcal{L}$ and $\mathcal{G}$ are pairwise independent function family, then for any q-query adaptive distinguisher D, it holds that*

$$\left|\Pr_{f \leftarrow \mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\Pi_{s,r},\Pi_{u,s},\Pi_{u,r})}[\mathsf{D}^f = 1] - \Pr_{\pi\leftarrow\Pi_{d,r}}[\mathsf{D}^\pi = 1]\right| \leq O\left(1/q^{c+1}\right),$$

for any $q \leq 2^{r-2}$.

In what follows we state two domain extension results (whose proofs are similar to that of Theorem 4.4). Recall that $\mathcal{ADW}_z$ makes two calls to $\mathcal{F}$, $2z$ calls to $\mathcal{M}$, and z calls to $\mathcal{Y}$ (in total $3z+2$ calls). In the first theorem (see Theorem 6.9) we rely on Corollary 6.7 and implement the function families $\Pi_{s,r}, \Pi_{u,s}$ and $\Pi_{u,r}$ using a single pseudorandom function family $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{s(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$. Assuming $u(n) \leq s(n)$ and $s(n) \leq r(n)$, the implementation is done in the natural way by padding with leading zeroes. This results with a PRF that makes $3z+2$ calls to the underlying PRF. In the second theorem (see Theorem 6.10) we take advantage of the fact that in Corollary 6.8 $u = 2$. This enables us to implement the function families $\mathcal{M}$ and $\mathcal{Y}$ as small tables of random values which are embedded into the PRF key. This results with a PRF that makes just two calls to the underlying PRF (but has a longer key).

Theorem 6.9. *Let d and q be integer functions, let $c > 1$ be a constant, let $z = 2(c+2)$, let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$, $\mathcal{L} = \{\mathcal{L} \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{r(n)}\}$ and $\mathcal{G} = \{\mathcal{G} \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{u(n)}\}$ be efficient pairwise independent function family ensembles, where $u(n) \leq s(n) \leq r(n)$, and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{s(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ be a (q,t,ε)-PRF. Then $\mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\mathcal{F},\mathcal{F},\mathcal{F}) = \{\mathcal{ADW}_z(\mathcal{H}_n,\mathcal{L}_n,\mathcal{G}_n,\mathcal{F}_n,\mathcal{F}_n,\mathcal{F}_n) \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ is a $(q, t - p\cdot q, (3z+2)\cdot\varepsilon + 1/q^{c+1})$-PRF, where $p \in$ poly is determined by the evaluation and sampling time of $\mathcal{H}$ and $\mathcal{F}$ and $q(n) \leq 2^{n-2}$.*

Theorem 6.10. *Let d and q be integer functions, let $c > 1$ be a constant, let $z = z(n) = 2(c+2)\cdot\log q(n)$, let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$, $\mathcal{L} = \{\mathcal{L} \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{s(n)}\}$ and $\mathcal{G} = \{\mathcal{G} \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{u(n)}\}$ be efficient pairwise independent function family ensembles, let $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$ (resp., $\mathcal{Y} = \{\mathcal{Y}_n\}_{n\in\mathbb{N}}$) be family of tables, such that $\mathcal{M}_n$ (resp., $\mathcal{Y}_n$) is a table of two (resp., $z(n)$) random elements from $\{0,1\}^{s(n)}$ (resp., $\{0,1\}^{r(n)}$), and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{s(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ be a (q,t,ε)-PRF. Then $\mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\mathcal{F},\mathcal{M},\mathcal{Y}) = \{\mathcal{ADW}_{z(n)}(\mathcal{H}_n,\mathcal{L}_n,\mathcal{G}_n,\mathcal{F}_n,\mathcal{M}_n,\mathcal{Y}_n) \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^{r(n)}\}_{n\in\mathbb{N}}$ is a $(q, t - p\cdot q, 2\varepsilon + 1/q^{c+1})$-PRF, where $p \in$ poly is determined by the evaluation and sampling time of $\mathcal{H}$ and $\mathcal{F}$ and $q(n) \leq 2^{n-2}$.*

Roughly speaking, the difference between Theorems 6.9 and 6.10 is that in Theorem 6.9 the resulting PRF makes a large constant (i.e., $6c+12$) number of queries to the underlying PRF, whereas in Theorem 6.10 it only makes two calls to the underlying PRF but the PRF key is longer (i.e., it has $(6c+12)\cdot\log q$ random values embedded into it).

6.2.1 Comparing the $\mathcal{PP}$ and $\mathcal{ADW}$ Based Constructions

Theorems 4.4, 6.9 and 6.10 present different tradeoffs between two types of resources: "cryptographic work" — the total evaluation time of the calls to the underlying short-domain PRF, and "combinatorial work" — the independence needed from the function families used.[16] In the $\mathcal{PP}$-based construction (Theorem 4.4), we minimize the number of calls to the PRF, thus keeping the cryptographic work small. But on the other hand, we require relatively high independence, thus requiring fairly much combinatorial work. In the first $\mathcal{ADW}$-based construction (Theorem 6.9), the situation is somewhat reversed: we minimize the independence needed, but make more calls to the PRF. In the second $\mathcal{ADW}$-based construction (Theorem 6.10), we minimize both the number of calls to the PRF and the independence needed, but we require much more hash functions, thus increasing again the combinatorial work.

[16] The independence affects the amount of random bits and evaluation time needed for these families.

In the following we instantiate Theorems 4.4, 6.9 and 6.10 in a specific setting, and compare their performance in terms of required randomness complexity, and evaluation time.[17] Our starting point is a length-preserving (q,t,ε)-PRF $\mathcal{F}=\{\mathcal{F}_n\}_{n\in\mathbb{N}}$. Our goal is to construct a function family with a larger domain, $\{0,1\}^{d(n)}$, whose security only deteriorates, comparing to that of $\mathcal{F}$, by an additive factor of $1/q(n)^c$, for $c:=c(n)>1$. Let $\mathsf{r}_{\mathcal{F}}(n)$ be the amount of random bits required to sample a random element in $\mathcal{F}_n$, and let $\mathsf{e}_{\mathcal{F}}(n)$ be the evaluation time of a single call to an element in $\mathcal{F}_n$. Let $\mathsf{e}^{(k)}_{\ell,n}$ be the evaluation time of a k-wise independent function family from $\{0,1\}^{\ell}$ to $\{0,1\}^n$. Recall that by Fact 2.6, sampling a random element in the latter function family requires $k\cdot\max\{\ell,n\}$ random bits.

Theorem 4.4 yields the following result, which makes only two calls to $\mathcal{F}$, but requires hash functions of high independence.

Corollary 6.11. *Let $c>1$ be an integer, let $\mathcal{H}=\{\mathcal{H}_n\colon\{0,1\}^{d(n)}\mapsto\{0,1\}^n\}_{n\in\mathbb{N}}$ be $\Omega(c\cdot\log q(n))$-wise independent function family, let $\mathcal{F}=\{\mathcal{F}_n\colon\{0,1\}^n\mapsto\{0,1\}^n\}_{n\in\mathbb{N}}$ be (q,t,ε)-PRF and let $\mathcal{PP}(\mathcal{H},\mathcal{F})=\{\mathcal{PP}(\mathcal{H}_n,\mathcal{F}_n)\colon\{0,1\}^{d(n)}\mapsto\{0,1\}^n\}_{n\in\mathbb{N}}$ be according to Definition 3.3. Then the following holds:*

1. $\left|\Pr_{f\leftarrow\mathcal{PP}(\mathcal{H}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1]-\Pr_{\pi\leftarrow\Pi_{d(n),n}}[\mathsf{D}^{\pi}(1^n)=1]\right|=O(\varepsilon(n)+1/q(n)^c)$, *for any adaptive $(t-p\cdot q)$-time q-query oracle-aided algorithm D and large enough $n\in\mathbb{N}$, where $p\in\mathrm{poly}$ is according to Theorem 4.4, and $q(n)\leq 2^{n-2}$.*
2. *The randomness complexity of $\mathcal{PP}(\mathcal{H},\mathcal{F})$ is $O(c\cdot d\cdot\log q)+2\cdot\mathsf{r}_{\mathcal{F}}$. Namely, there exists an algorithm that on input 1^n, uses $O(c\cdot d(n)\cdot\log q(n))+2\cdot\mathsf{r}_{\mathcal{F}}(n)$ random bits, and outputs a random element in $\mathcal{PP}(\mathcal{H}_n,\mathcal{F}_n)$.*
3. *The evaluation time of $\mathcal{PP}(\mathcal{H},\mathcal{F})$ is $O\left(\mathsf{e}^{(c\cdot\log q)}_{d,n}\right)+2\cdot\mathsf{e}_{\mathcal{F}}$. Namely, there exists an algorithm that on input $f\in\mathcal{PP}(n)$ and $x\in\{0,1\}^{d(n)}$, runs in time $O\left(\mathsf{e}^{(c\cdot\log q(n))}_{d(n),n}\right)+2\cdot\mathsf{e}_{\mathcal{F}}(n)$, and outputs $f(x)$.*

In comparison, Theorem 6.9 allows us to reduce the independence needed, in the price of increasing the number of calls to $\mathcal{F}$.

Corollary 6.12. *Let $c>1$ be an integer, let $z=2(c+2)$, let $\mathcal{H}=\{\mathcal{H}_n\colon\{0,1\}^{d(n)}\mapsto\{0,1\}^n\}_{n\in\mathbb{N}}$ be efficient pairwise independent function family ensembles, let $\mathcal{F}=\{\mathcal{F}_n\colon\{0,1\}^n\mapsto\{0,1\}^n\}_{n\in\mathbb{N}}$ be a (q,t,ε)-PRF and let $\mathcal{ADW}_z(\mathcal{H},\mathcal{F}):=\mathcal{ADW}_z(\mathcal{H},\mathcal{H},\mathcal{H},\mathcal{F},\mathcal{F},\mathcal{F})=\{\mathcal{ADW}_z(\mathcal{H}_n,\mathcal{F}_n):=\mathcal{ADW}_z(\mathcal{H}_n,\mathcal{H}_n,\mathcal{H}_n,\mathcal{F}_n,\mathcal{F}_n,\mathcal{F}_n)\colon\{0,1\}^{d(n)}\mapsto\{0,1\}^n\}_{n\in\mathbb{N}}$ be according to Definition 6.1.*

1. $\left|\Pr_{f\leftarrow\mathcal{ADW}_z(\mathcal{H}_n,\mathcal{F}_n)}[\mathsf{D}^f(1^n)=1]-\Pr_{\pi\leftarrow\Pi_{d(n),n}}[\mathsf{D}^{\pi}(1^n)=1]\right|=O(c\cdot\varepsilon(n)+1/q(n)^c)$, *for any $(t-p\cdot q)$-time q-query adaptive oracle-aided algorithm D and large enough $n\in\mathbb{N}$, where $p\in\mathrm{poly}$ is according to Theorem 6.9, and $q(n)\leq 2^{n-2}$.*
2. *The randomness complexity of $\mathcal{ADW}_z(\mathcal{H},\mathcal{F})$ is $O(c\cdot d)+(6c+14)\cdot\mathsf{r}_{\mathcal{F}}$. Namely, there exists an algorithm that on input 1^n, uses $O(c\cdot d(n))+(6c+14)\cdot\mathsf{r}_{\mathcal{F}}(n)$ random bits, and outputs a random element in $\mathcal{ADW}_z(\mathcal{H}_n,\mathcal{F}_n)$.*

[17] Another criterion of comparison is the sampling time. In our settings it is analogous to the evaluation time, so we omit it.

Function Family	Randomness Complexity (Key Size)	Evaluation Time
$\mathcal{PP}$ (Corollary 6.11)	$O(c \cdot d \cdot \log q) + 2 \cdot \mathsf{r}_{\mathcal{F}}$	$O\left(\mathsf{e}_{d,n}^{(c \cdot \log q)}\right) + 2 \cdot \mathsf{e}_{\mathcal{F}}$
$\mathcal{ADW}$ (Corollary 6.12)	$O(c \cdot d) + (6c+14) \cdot \mathsf{r}_{\mathcal{F}}$	$O\left(c \cdot \mathsf{e}_{d,n}^{(2)}\right) + (6c+14) \cdot \mathsf{e}_{\mathcal{F}}$
$\mathcal{ADW}$ (Corollary 6.13)	$O(c \cdot d \cdot \log q) + 2 \cdot \mathsf{r}_{\mathcal{F}}$	$O\left(c \cdot \log q \cdot \mathsf{e}_{d,n}^{(2)}\right) + 2 \cdot \mathsf{e}_{\mathcal{F}}$

Table 1: Comparison between the domain extension results based on the function family $\mathcal{PP}$ instantiated as in Corollary 6.11 and the function family $\mathcal{ADW}$ instantiated as in Corollaries 6.12 and 6.13.

3. *The evaluation time of* $\mathcal{ADW}_z(\mathcal{H},\mathcal{F})$ *is* $O\left(c \cdot \mathsf{e}_{d,n}^{(2)}\right) + (6c+14) \cdot \mathsf{e}_{\mathcal{F}}$. *Namely, there exists an algorithm that on input* $f \in \mathcal{ADW}_z(\mathcal{H}_n,\mathcal{F}_n)$ *and* $x \in \{0,1\}^{d(n)}$, *runs in time* $O\left(c \cdot \mathsf{e}_{d(n),n}^{(2)}\right) + (6c+14) \cdot \mathsf{e}_{\mathcal{F}}(n)$, *and outputs* $f(x)$.

Finally, Theorem 6.10 allows us to make only two calls to $\mathcal{F}$ and keep both the independence needed low, in the price of needing much more hash functions.

Corollary 6.13. *Let* $c > 1$ *be an integer, let* q *be an integer function, let* $z(n) = 2(c+2) \cdot \log q(n)$, *let* $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ *be efficient pairwise independent function family ensembles, let* $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$ *(resp.,* $\mathcal{Y} = \{\mathcal{Y}_n\}_{n\in\mathbb{N}}$*) be a family of tables, where* $\mathcal{M}_n$ *(resp.,* $\mathcal{Y}_n$*) is a random table of two (resp., z) elements from* $\{0,1\}^n$, *let* $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ *be a* (q,t,ε)*-PRF and let* $\mathcal{ADW}_z(\mathcal{H},\mathcal{F},\mathcal{M},\mathcal{Y}) := \mathcal{ADW}_z(\mathcal{H},\mathcal{H},\mathcal{H},\mathcal{F},\mathcal{M},\mathcal{Y}) = \{\mathcal{ADW}_{z(n)}(\mathcal{H}_n,\mathcal{F}_n,\mathcal{M}_n,\mathcal{Y}_n) := \mathcal{ADW}_{z(n)}(\mathcal{H}_n,\mathcal{H}_n,\mathcal{H}_n,\mathcal{F}_n,\mathcal{M}_n,\mathcal{Y}_n) \colon \{0,1\}^{d(n)} \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ *be according to Definition 6.1.*

1. $\left|\Pr_{f\leftarrow\mathcal{ADW}_{z(n)}(\mathcal{H}_n,\mathcal{F}_n,\mathcal{M}_n,\mathcal{Y}_n)}[\mathsf{D}^f(1^n)=1] - \Pr_{\pi\leftarrow\Pi_{d(n),n}}[\mathsf{D}^\pi(1^n)=1]\right| = O(\varepsilon(n) + 1/q(n)^c)$, *for any* $(t - p\cdot q)$*-time* q*-query adaptive oracle-aided algorithm* D *and large enough* $n \in \mathbb{N}$, *where* $p \in \mathrm{poly}$ *is according to Theorem 6.9, and* $q(n) \leq 2^{n-2}$.

2. *The randomness complexity of* $\mathcal{ADW}_z(\mathcal{H},\mathcal{F},\mathcal{M},\mathcal{Y})$ *is* $O(c \cdot d \cdot \log q) + 2 \cdot \mathsf{r}_{\mathcal{F}}$. *Namely, there exists an algorithm that on input* 1^n, *uses* $O(c \cdot d(n) \cdot \log q(n)) + 2 \cdot \mathsf{r}_{\mathcal{F}}(n)$ *random bits, and outputs a random element in* $\mathcal{ADW}_{z(n)}(\mathcal{H}_n,\mathcal{F}_n,\mathcal{M}_n,\mathcal{Y}_n)$.[18]

3. *The evaluation time of* $\mathcal{ADW}_z(\mathcal{H},\mathcal{F},\mathcal{M},\mathcal{Y})$ *is* $O\left(c \cdot \log q \cdot \mathsf{e}_{d,n}^{(2)}\right) + 2 \cdot \mathsf{e}_{\mathcal{F}}$. *Namely, there exists an algorithm that on input* $f \in \mathcal{ADW}_{z(n)}(\mathcal{H}_n,\mathcal{F}_n,\mathcal{M}_n,\mathcal{Y}_n)$ *and* $x \in \{0,1\}^{d(n)}$, *runs in time* $O\left(c \cdot \log q(n) \cdot \mathsf{e}_{d(n),n}^{(2)}\right) + 2 \cdot \mathsf{e}_{\mathcal{F}}(n)$, *and outputs* $f(x)$.

The above corollaries are summarized in Table 1.

6.3 From Non-Adaptive to Adaptive PRF Via the $\mathcal{ADW}$ Family

In this subsection we present a non-adaptive to adaptive transformation using the Aumüller et al. [4] family, $\mathcal{ADW}$. As in the previous section, this allows us to avoid the large independence required for using the function family $\mathcal{PP}$ of Pagh and Pagh [42]. For simplicity we state only the reduction

[18] The constant hidden in the big O notation in this item is slightly larger than the corresponding constant in the same item in Corollary 6.12. The difference comes from including $\mathcal{M}$ and $\mathcal{Y}$ into the key.

follows Corollary 6.6. Corollary 6.5 also yields non-adaptive to adaptive reduction, with different tradeoff between the randomness complecity (key size) of the PRF to its evaluation time (see Section 6.2.1).

Recall that our reduction from Section 5 requires that all calls to the PRF $\mathcal{F}$ are within the first $4q(n)$ first elements of $\{0,1\}^n$. We make sure the above holds by the following settings. Specifically, Corollary 6.6 yields the following theorem with respect to the above function family (similarly to Theorem 5.4).

Theorem 6.14. *Let q be integer functions, let $c > 1$ be a constant, let $z = z(n) = 2(c+2)\cdot\log q(n)$, let $\mathcal{H} = \{\mathcal{H}_n\colon \{0,1\}^n \mapsto [4q(n)]_{\{0,1\}^n}\}$, $\mathcal{L} = \{\mathcal{L}\colon \{0,1\}^n \mapsto \{0,1\}^n\}$ and $\mathcal{G} = \{\mathcal{G}\colon \{0,1\}^n \mapsto \{0,1\}\}$ be efficient pairwise independent function family ensembles, let $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$ (resp., $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$) be family of tables, such that $\mathcal{M}_n$ (resp., $\mathcal{Y}_n$) is a random table of two (resp., $z(n)$) elements from $[4q(n)]_{\{0,1\}^n}$ (resp., $\{0,1\}^n$), and let $\mathcal{F} = \{\mathcal{F}_n\colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be a* non-adaptive *$(4q, p\cdot t, \varepsilon)$-PRF, where $p \in$ poly is determined by the evaluation time of $q, \mathcal{H}, \mathcal{G}$, $\mathcal{L}$ and $\mathcal{F}$.*

Then, $\mathcal{ADW}_z(\mathcal{H},\mathcal{L},\mathcal{G},\mathcal{F},\mathcal{M},\mathcal{Y}) = \{\mathcal{ADW}_z(n)(\mathcal{H}_n,\mathcal{L}_n,\mathcal{G}_n,\mathcal{F}_n,\mathcal{M}_n,\mathcal{Y}_n)\colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ is a $(q, t, 2\varepsilon + 1/q^{c+1})$-PRF.

7 Further Research

The focus of this paper is on PRFs. Specifically, in Sections 4 to 6 we have shown domain extension techniques and non-adaptive to adaptive transformations for PRFs that provide a nice tradeoff between combinatorial work, cryptographic work and error. In general, hardness-preserving reductions between pseudorandom objects has led to fruitful research with many result (some of which we review next). It is an interesting question whether our technique has any bearing on other models.

Perhaps the most interesting model for this kind of reductions is pseudorandom permutations (PRPs) (without going through a PRP-to-PRF reduction). Given a family of (q, t, ε)-PRPs from n-bits to n bits, how can we construct a family of PRPs with *larger* domain while preserving its security? How about constructing a family of PRPs with *smaller* domain while preserving its security? Finally, it is also interesting how to transform a family of (q, t, ε)-PRPs that is secure against non-adaptive adversaries to a family of PRPs that are also secure against adaptive adversaries (see discussion in Section 1.4). One related paper is that of Hoang et al. [24], that gives a method to convert a PRF into a PRP with beyond-birthday security (see also [50, 35]). Another related work is of Håstad [22] that showed how to extend the domain of a PRP.

A different model of interest is message authentication codes (MACs). In this model, we are interested in designing domain extension techniques that given an n-bit to n-bit MAC with MAC security ε against q queries provide variable-length MAC with some (good enough) promise on the MAC security in terms of q and ε. The best answer to-date for this question was given by Dodis and Steinberger [16] that showed that given an n-bit to n-bit MAC with MAC security ε against q queries, it is possible to get a variable-length MAC achieving MAC security $O(\varepsilon \cdot q \cdot \text{poly}(n))$ against queries of total length qn.

Another interesting model is public random functions. A public random function $f : \{0,1\}^m \to \{0,1\}^n$ is a system with a public and private interface which behaves as the same random function at both interfaces. In other words, a public random function can be interpreted as a random oracle. In this model, again, the domain extension problem is very interesting. To date, the best

construction is of Maurer and Tessaro [34] that presented a construction $\mathbf{C}_{\varepsilon,m,\ell}$ that extends public random functions $f : \{0,1\}^n \to \{0,1\}^n$ to a function $\mathbf{C}_{\varepsilon,m,\ell}(f) : \{0,1\}^{m(n)} \to \{0,1\}^{\ell(n)}$ with time complexity $\text{poly}(n, 1/\varepsilon)$ and which is secure against adversaries which make up to $\Theta(2^{(1-\varepsilon)n})$ queries.

On a different note, notice that all of our constructions assume that we are given the number of queries *in advance*. Can we get any non-trivial results given only an *upper bound* on the number of queries (in any of the models above)? In particular, in the non-adaptive to adaptive reduction of Section 5 we assumed that $q(n)$, the number of queries to which the non adaptive function is resistent to, is known. What can be done if it is not known? This issue was addressed by Berman and Haitner [7] in a non security preserving manner.

Recently Pătraşcu and Thorup [49] have shown that for many data structure problems it is possible to use tabulation hashing even though it is 'merely' 3-wise independent. The question is whether this has any bearing on cryptographic constructions.

8 Acknowledgments

We thank Eylon Yogev and the anonymous referees for their helpful comments. The third author would like to thank his M.Sc advisor Ran Raz for his support.

References

[1] W. Aiello and R. Venkatesan. Foiling birthday attacks in length-doubling transformations - benes: A non-reversible alternative to feistel. In *Advances in Cryptology – EUROCRYPT '96*, pages 307–320, 1996.

[2] Y. Arbitman, M. Naor and G. Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *Proceedings of the 51th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 787–796, 2010.

[3] M. Aumüller, M. Dietzfelbinger and P. Woelfel. Explicit and efficient hash families suffice for cuckoo hashing with a stash. In L. Epstein and P. Ferragina, editors, *ESA*, volume 7501 of *Lecture Notes in Computer Science*, pages 108–120. Springer, 2012.

[4] M. Aumüller, M. Dietzfelbinger and P. Woelfel. Explicit and efficient hash families suffice for cuckoo hashing with a stash. *Algorithmica*, 70(3):428–456, 2014.

[5] M. Bellare and S. Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In *Advances in Cryptology – CRYPTO '89*, pages 194–211, 1989.

[6] M. Bellare, O. Goldreich and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In *Advances in Cryptology – CRYPTO '99*, pages 270–287, 1999.

[7] I. Berman and I. Haitner. From non-adaptive to adaptive pseudorandom functions. *J. Cryptology*, 28(2):297–311, 2015.

[8] I. Berman, I. Haitner, I. Komargodski and M. Naor. Hardness preserving reductions via cuckoo hashing. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013*, pages 40–59, 2013.

[9] O. Billet, J. Etrog and H. Gilbert. Lightweight privacy preserving authentication for rfid using a stream cipher. In *18th International Symposium on the Foundations of Software Engineering (FSE)*, pages 55–74, 2010.

[10] M. Blum, W. S. Evans, P. Gemmell, S. Kannan and M. Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[11] L. J. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, pages 143–154, 1979.

[12] N. Chandran and S. Garg. Balancing output length and query bound in hardness preserving constructions of pseudorandom functions. In *Progress in Cryptology - INDOCRYPT 2014*, pages 89–103, 2014.

[13] B. Chor, A. Fiat, M. Naor and B. Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

[14] M. Dietzfelbinger and C. Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theor. Comput. Sci.*, 380(1-2):47–68, 2007.

[15] M. Dietzfelbinger and P. Woelfel. Almost random graphs with simple hash functions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 629–638, 2003.

[16] Y. Dodis and J. P. Steinberger. Domain extension for macs beyond the birthday barrier. In *Advances in Cryptology – EUROCRYPT 2011*, pages 323–342, 2011.

[17] D. Fotakis, R. Pagh, P. Sanders and P. G. Spirakis. Space efficient hash tables with worst case constant access time. *Theory Comput. Syst.*, 38(2):229–248, 2005.

[18] A. M. Frieze, P. Melsted and M. Mitzenmacher. An analysis of random-walk cuckoo hashing. *SIAM J. Comput.*, 40(2):291–308, 2011.

[19] O. Goldreich. Towards a theory of software protection. In *Advances in Cryptology – CRYPTO '86*, pages 426–439, 1986.

[20] O. Goldreich, S. Goldwasser and S. Micali. On the cryptographic applications of random functions. In *Advances in Cryptology – CRYPTO '84*, pages 276–288, 1984.

[21] O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, pages 792–807, 1986.

[22] J. Håstad. The square lattice shuffle. *Random Structures & Algorithms*, 29(4):466–474, 2006.

[23] J. Håstad, R. Impagliazzo, L. A. Levin and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, pages 1364–1396, 1999.

[24] V. T. Hoang, B. Morris and P. Rogaway. An enciphering scheme based on a card shuffle. In *Advances in Cryptology – CRYPTO 2012*, pages 1–13, 2012.

[25] A. Jain, K. Pietrzak and A. Tentes. Hardness preserving constructions of pseudorandom functions. In *Theory of Cryptography, 9th Theory of Cryptography Conference, TCC 2012*, pages 369–382, 2012.

[26] D. Jetchev, O. Özen and M. Stam. Understanding adaptivity: Random systems revisited. In *Advances in Cryptology – ASIACRYPT 2012*, pages 313–330, 2012.

[27] E. Kaplan, M. Naor and O. Reingold. Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009.

[28] A. Kirsch, M. Mitzenmacher and U. Wieder. More robust hashing: Cuckoo hashing with a stash. *SIAM J. Comput.*, 39(4):1543–1561, 2009.

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

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

[31] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[32] U. M. Maurer. Indistinguishability of random systems. In *Advances in Cryptology – EUROCRYPT 2002*, pages 110–132, 2002.

[33] U. M. Maurer and K. Pietrzak. Composition of random systems: When two weak make one strong. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 410–427, 2004.

[34] U. M. Maurer and S. Tessaro. Domain extension of public random functions: Beyond the birthday barrier. In *Advances in Cryptology – CRYPTO 2007*, pages 187–204, 2007.

[35] B. Morris and P. Rogaway. Sometimes-recurse shuffle - almost-random permutations in logarithmic expected time. In *Advances in Cryptology – EUROCRYPT 2014*, pages 311–326, 2014.

[36] S. Myers. Black-box composition does not imply adaptive security. In *Advances in Cryptology – EUROCRYPT 2004*, pages 189–206, 2004.

[37] M. Nandi. A unified method for improving prf bounds for a class of blockcipher based macs. In *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea*, pages 212–229, 2010.

[38] M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-rackoff revisited. *Journal of Cryptology*, 12(1):29–66, 1999.

[39] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999.

[40] M. Naor and O. Reingold. Constructing pseudo-random permutations with a prescribed structure. *Journal of Cryptology*, 15(2):97–102, 2002.

[41] R. Ostrovsky. An efficient software protection scheme. In *Advances in Cryptology – CRYPTO '89*, 1989.

[42] A. Pagh and R. Pagh. Uniform hashing in constant time and optimal space. *SIAM Journal on Computing*, 38(1):85–96, 2008.

[43] R. Pagh and F. F. Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.

[44] J. Patarin. Security of random feistel schemes with 5 or more rounds. In *Advances in Cryptology – CRYPTO 2004*, pages 106–122, 2004.

[45] J. Patarin. A proof of security in $o(2^n)$ for the benes scheme. In *Progress in Cryptology - AFRICACRYPT 2008*, pages 209–220, 2008.

[46] J. Patarin. Security of balanced and unbalanced feistel schemes with linear non equalities. *IACR Cryptology ePrint Archive*, 2010:293, 2010.

[47] K. Pietrzak. Composition does not imply adaptive security. In *Advances in Cryptology – CRYPTO 2005*, pages 55–65, 2005.

[48] K. Pietrzak. Composition implies adaptive security in minicrypt. In *Advances in Cryptology – EUROCRYPT 2006*, pages 328–338, 2006.

[49] M. Pătraşcu and M. Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3):14, 2012.

[50] T. Ristenpart and S. Yilek. The mix-and-cut shuffle: Small-domain encryption secure against N queries. In *Advances in Cryptology – CRYPTO 2013*, pages 392–409, 2013.

[51] A. Siegel. On universal classes of extremely random constant-time hash functions. *SIAM Journal on Computing*, 33(3):505–543, 2004.

[52] M. N. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

A Proof of Lemma 3.2

In this section we prove Lemma 3.2 from Section 3. We mentioned again that Lemma 3.2 can be derived as a special case of a result given in [26, Theorem 12] (closing a gap in the proof appearing in [32]). Yet, for the sake of completeness, we include an independent proof of this lemma below.

Definition A.1 (Restating Definition 3.1)**.** *Let $\mathcal{S}$ and $\mathcal{T}$ be sets. A set $\mathcal{M} \subseteq \mathcal{S}^* \times \mathcal{T}$ is* left-monotone*, if for every $(\overline{s_1}, t) \in \mathcal{M}$ and every $\overline{s_2} \in \mathcal{S}^*$ that has $\overline{s_1}$ as a prefix, it holds that $(\overline{s_2}, t) \in \mathcal{M}$.*

Lemma A.2 (Restating Lemma 3.2)**.** *Let $\mathcal{U}$ and $\mathcal{V}$ be non-empty sets, let $\mathcal{F} = \mathcal{F}(\mathcal{U},\mathcal{V}) = \{f_{u,v}\colon \mathcal{D} \mapsto \mathcal{R}\}_{(u,v)\in\mathcal{U}\times\mathcal{V}}$ be a function family and let* BAD $\subseteq \mathcal{D}^* \times \mathcal{U}$ *be left monotone. Let $t \in \mathbb{N}$, and assume that for every $\overline{q} = (q_1,\ldots,q_{|\overline{q}|}) \in \mathcal{D}^{\leq t}$:*[19]

1. $\left(f(q_1),\ldots,f(q_{|\overline{q}|})\right)_{f\leftarrow\{f_{u,v}\colon v\in\mathcal{V}\}}$ *is uniform over $\mathcal{R}^{|\overline{q}|}$, for every $u \in \mathcal{U}$ with $(\overline{q},u) \notin$* BAD*, and*
2. $\Pr_{u\leftarrow\mathcal{U}}[(\overline{q},u) \in \text{BAD}] \leq \varepsilon$.

Then

$$\left|\Pr_{\substack{u\leftarrow\mathcal{U}\\ v\leftarrow\mathcal{V}}}[\mathsf{D}^{f_{u,v}} = 1] - \Pr_{\pi\leftarrow\Pi}[\mathsf{D}^{\pi} = 1]\right| \leq \varepsilon,$$

for every t-query oracle-aided adaptive *algorithm* D*, letting* Π *be the set of all functions from $\mathcal{D}$ to $\mathcal{R}$.*

Proof. Let D be an t-query distinguisher. We assume for simplicity that D is deterministic (the reduction to the randomized case is standard) and makes exactly t valid (i.e., inside $\mathcal{D}$) distinct queries. To prove the lemma we consider a process (Algorithm A.3) that runs D twice: one giving it completely random answers and the second time, choosing a $u \leftarrow \mathcal{U}$ and continuing answering with the same answers as the first round until we hit a BAD event according to the queries and the chosen u. We then choose a random v that is consistent with answers given so far and continue answering with it.

Intuitively, the answers provided to D in the first round are distributed like the answers D expects to get from a *truly random function*, while the answers provided to D in the second round are distributed like the answers D expects to get from a random function in $\mathcal{F}$. But, since these answers are the same until a BAD event occurs, the distinguishing ability of D is bounded by the probability of such an event to occur. Since, u is chosen *after* D has already "committed" to the queries it is going to make, this probability is bounded by the non-adaptive property of $\mathcal{F}$.

For a vector $\overline{v} = (v_1,\ldots,v_t)$, let $\overline{v}_{1,\ldots,i}$ be the first i element in $\overline{v}$ (i.e., $\overline{v}_{1,\ldots,i} = (v_1,\ldots,v_i)$) and let $\overline{v}_{1,\ldots,0} = \lambda$, where λ is the empty vector. Consider the following random process:

Algorithm A.3.

1. *Emulate* D*, while answering the i^{th} query q_i with $a_i \leftarrow \mathcal{R}$.*
   Set $\overline{q} = (q_1,\ldots,q_t)$ and $\overline{a} = (a_1,\ldots,a_t)$.
2. *Choose $u \leftarrow \mathcal{U}$ and set $v = \perp$.*
3. *If $(\lambda,u) \in$* BAD*, set $v \leftarrow \mathcal{V}$.*
4. *Emulate* D *again, while answering the i^{th} query q'_i according to the following procedure:*
   (a) *If $(\overline{q}'_{1,\ldots,i} = (q'_1,\ldots,q'_i),u) \notin$* BAD*, answer with $a'_i = a_i$ (the same a_i from Step 1).*
   (b) *Otherwise ($(\overline{q}'_{1,\ldots,i},u) \in$* BAD*):*
      i. *If $v = \perp$, set $v \leftarrow \{v' \in \mathcal{V}\colon \forall j \in [i-1]\colon f_{u,v'}(q'_j) = a'_j\}$.*
      ii. *Answer with $a'_i = f_{u,v}(q'_i)$.*

[19] Recall that for a set $\mathcal{S}$ and an integer t, $\mathcal{S}^{\leq t}$ denotes the set $\{\overline{s} \in \mathcal{S}^*\colon\ |\overline{s}| \leq t \ \ \wedge \ \ \overline{s}[i] \neq \overline{s}[j] \ \ \forall i \neq j \in [|\overline{s}|]\}$.

5. *Set* $\overline{q'} = (q'_1, \ldots, q'_t)$ *and* $\overline{a'} = (a'_1, \ldots, a'_t)$. *In case* $v = \perp$, *set* $v \leftarrow \{v' \in \mathcal{V} \colon \forall j \in [t] \colon f_{u,v'}(q'_j) = a'_j\}$.

Let $\overline{A}$, $\overline{Q}$, $\overline{A'}$, $\overline{Q'}$, U and V be the (jointly distributed) random variables induced by the values of $\overline{q}$, $\overline{a}$, $\overline{q'}$, $\overline{a'}$, u and v respectively, in a random execution of Algorithm A.3. By definition $\overline{A}$ has the same distribution as the oracle answers in a random execution of D^{π} with $\pi \leftarrow \Pi$. In Claim A.4 we show that $\overline{A}'$ is distributed the same as the oracle answers in a random execution of $\mathsf{D}^{f_{u,v}}$ with $(u, v) \leftarrow \mathcal{U} \times \mathcal{V}$. Using it, we now conclude the proof by bounding the statistical distance between $\overline{A}$ and $\overline{A}'$.

Since the queries and answers in both emulations of D at Algorithm A.3 are the same until $(\overline{Q}_{1,\ldots,i}, U) \in \mathrm{BAD}$ for some $i \in [t]$, and since BAD is monotone, it holds that

$$\Pr[\overline{A} \neq \overline{A}'] \leq \Pr[(\overline{Q}, U) \in \mathrm{BAD}] \tag{5}$$

In addition, since U is chosen *after* $\overline{Q}$, the second condition of Lemma A.2 yields that

$$\Pr[(\overline{Q}, U) \in \mathrm{BAD}] \leq \varepsilon \tag{6}$$

It follows that $\Pr[\overline{A} \neq \overline{A}'] \leq \varepsilon$ and therefore $\mathrm{SD}(\overline{A}, \overline{A}') \leq \varepsilon$.

We conclude that

$$\left| \Pr_{\substack{u \leftarrow \mathcal{U} \\ v \leftarrow \mathcal{V}}}[\mathsf{D}^{f_{u,v}} = 1] - \Pr_{\pi \leftarrow \Pi}[\mathsf{D}^{\pi} = 1] \right| \leq \mathrm{SD}(\overline{A}, \overline{A}') \leq \varepsilon.$$

□

Claim A.4. $\overline{A}'$ *has the same distribution as the oracle answers in a random execution of* $\mathsf{D}^{f_{u,v}}$ *with* $(u, v) \leftarrow \mathcal{U} \times \mathcal{V}$.

Proof. It is easy to verify that $\overline{A}'$ is the oracle answers in $D^{f_{U,V}}$. Hence, to obtain the claim we need to show that (U, V) is uniformly distributed over $\mathcal{U} \times \mathcal{V}$. The definition of Algorithm A.3 assures that U is uniformly distributed over $\mathcal{U}$, so it is left to show that conditioned on any fixing u of U, the value of V is uniformly distributed over $\mathcal{V}$.

In the following we condition on $U = u \in \mathcal{U}$. For an answers vector $\overline{w} \in \mathcal{R}^k$, let $\overline{q}_{\overline{w}}$ [resp., $\overline{q}^{+}_{\overline{w}}$] be the first k [resp., $k+1$] queries asked by D, assuming that it gets $\overline{w}$ as the first k answers (since D is deterministic these values are well defined). Let $\mathcal{S}_{\overline{w}} = \{v \in \mathcal{V} \colon f_{u,v}(\overline{q}_{\overline{w}}) = \overline{w}\}$ and let $W = \{\overline{w} \in \mathcal{R}^* \colon |\overline{w}| \leq t \wedge (\overline{q}_{\overline{w}}, u) \notin \mathrm{BAD}\}$. If $\lambda \notin W$, it follows that $(\lambda, u) \in \mathrm{BAD}$, and thus Algorithm A.3 chooses v at Step 3. Hence V is uniformly distributed over $\mathcal{V}$. In case $\lambda \in W$, we conclude the proof by applying the following claim (proven below) with $\overline{w} = \lambda$ (note that $\mathcal{S}_{\lambda} = \mathcal{V}$).

Claim A.5. *Conditioned on* $\overline{A'}_{1,\ldots,i} = \overline{w} \in W$ *for some* $i \in \{0, , \ldots, t\}$, *the value of* V *is uniformly distributed over* $\mathcal{S}_{\overline{w}}$.

□

Proof of Claim A.5. We prove by reverse induction on $i = |\overline{w}|$. For the base case $i = t$, we note that (by definition) Algorithm A.3 chooses v at Step 5, and thus V is uniformly distributed over $\mathcal{S}_{\overline{w}}$. In the following we assume the hypothesis holds for $i+1$, and condition on $\overline{A'}_{1,\ldots,i} = \overline{w} \in W$. In case $(\overline{q}^{+}_{\overline{w}}, u) \in \mathrm{BAD}$, Algorithm A.3 chooses v at Step 4(b)i, and thus V is uniformly distributed over $\mathcal{S}_{\overline{w}}$. So it is left to handle the case $(\overline{q}^{+}_{\overline{w}}, u) \notin \mathrm{BAD}$.

Fix $v' \in \mathcal{S}_{\overline{w}}$ and let $a \in \mathcal{R}$ be such that $v' \in \mathcal{S}_{\overline{w}\circ a}$, where '$\circ$' denotes vector concatenation (i.e., for $\overline{w} = (w_1, \ldots, w_i)$, $\overline{w} \circ a = (w_1, \ldots, w_i, a)$). Conditioning on $A'_{i+1} = a$, we can apply the induction hypothesis on $\overline{w} \circ a$ (since $\overline{w} \circ a \in W$) to get that V is uniformly distributed over $\mathcal{S}_{\overline{w}\circ a}$. It follows that

$$\begin{aligned}\Pr[V = v' \mid \overline{A'}_{1,\ldots,i} = \overline{w}] &= \Pr[A'_{i+1} = a \mid \overline{A'}_{1,\ldots,i} = \overline{w}] \cdot \Pr[v = v' \mid \overline{A'}_{1,\ldots,i+1} = \overline{w} \circ a] \\ &= \frac{1}{|R|} \cdot \frac{1}{|\mathcal{S}_{\overline{w}\circ a}|} \\ &= \frac{1}{|R|} \cdot \frac{|\mathcal{R}|^{|\overline{w}|+1}}{|\mathcal{V}|} \\ &= \frac{|\mathcal{R}|^{|\overline{w}|}}{|\mathcal{V}|} = \frac{1}{|\mathcal{S}_{\overline{w}}|},\end{aligned}$$

concluding the induction step. The second equality holds by the induction hypothesis, and for the third one we note that

$$\frac{|\mathcal{S}_{\overline{w}'}|}{|\mathcal{V}|} = \Pr_{v\leftarrow\mathcal{V}}[v \in \mathcal{S}_{\overline{w}'}] = \Pr_{v\leftarrow\mathcal{V}}[f_{u,v}(\overline{q}_{\overline{w}'}) = \overline{w}'] = \frac{1}{|\mathcal{R}|^{|\overline{w}'|}}, \tag{7}$$

for every $\overline{w}' \in W$, where the third equality of Equation (7) holds by the first property of $\mathcal{F}(\mathcal{U}, \mathcal{V})$ (as stated in Lemma A.2). □

B Hardness-Preserving PRG to PRF Reductions

Another application of our technique is a hardness-preserving construction of PRFs from pseudorandom generators (PRGs). For instance, constructing $2^{c'n}$-PRF for some $0 < c' < c$, from a 2^{cn}-PRG. The efficiency of such constructions is measured by the number of calls made to the underlying PRG as well as other parameters such as representation size.

The construction of Goldreich et al. [21] (i.e., $\mathcal{GGM}$) is in fact hardness preserving according to the above criterion. Their construction, however, makes n calls to the underlying PRG, which might be too expensive in some settings.

Proposition B.1 ([21]). *Let G be a length-doubling (t, ε)-PRG whose evaluation time is e_G. For any efficiently-computable integer functions m and ℓ, there exists an efficient oracle-aided function family ensemble whose n'th function family, denoted $\mathcal{GGM}^G_{m(n)\to\ell(n)}$, maps strings of length $m(n)$ to strings of length $\ell(n)$, makes $m(n)$ calls to G and is a $(q, t - m \cdot q \cdot e_G(\ell), m \cdot q \cdot \varepsilon(\ell))$-PRF for any integer function q.*[20]

As already mentioned in the introduction, in order to reduce the number of calls to the underlying PRG, Levin [29] suggested to first hash the input to a smaller domain, and only then apply

[20] $\mathcal{GGM}^G_{m(n)\to\ell(n)}$ is a variant of the standard $\mathcal{GGM}$ function family, that on input of length $m(n)$ uses seed of length $\ell(n)$ for the underlying generator, rather than seed of length $m(n)$. Formally, $\mathcal{GGM}^G_{m\to\ell}$ is the function family ensemble $\{\mathcal{GGM}^G_{m(n)\to\ell(n)}\}_{n\in\mathbb{N}}$, where $\mathcal{GGM}^G_{m(n)\to\ell(n)} = \{f_r\}_{r\in\{0,1\}^{\ell(n)}}$, and for $r \in \{0,1\}^{\ell(n)}$, the oracle-aided function $f_r \colon \{0,1\}^{m(n)} \mapsto \{0,1\}^{\ell(n)}$ is defined as follows: given oracle access to a length-doubling function G and input $x \in \{0,1\}^{m(n)}$, $f^G_r(x) = r_x$, where r_x is recursively defined by $r_\varepsilon = r$, and, for a string w, $r_{w||0}||r_{w||1} = G(r_w)$. The original $\mathcal{GGM}$ construction was length-preserving, i.e., $m(n) = \ell(n) = n$.

$\mathcal{GGM}$ (this is known as "Levin's trick"). The resulting construction, however, is not hardness preserving due to the "birthday attack" described in Section 1.

While the $\mathcal{GGM}$ construction seems optimal for the security it achieves (as shown in [25]), in some settings the number of queries the distinguisher can make is *strictly less* than its running time. Consider a distinguisher of running time 2^{cn} that can only make $2^{\sqrt{n}} \ll 2^{cn}$ queries. In such settings the security of the $\mathcal{GGM}$ construction seems like an overkill and raises the question of whether there exist more efficient reductions. Jain et al. [25] (who raised the above question) gave the following partial answer, by designing a domain extension method tailored to PRG to PRF reductions for a specific range of parameters.

Theorem B.2 ([25])**.** *Let G be a length-doubling 2^{cn}-PRG. Let $c > 0$, $1/2 \leq \alpha < 1$ and $q(n) = 2^{n^\alpha}$. There exists a length-preserving function family $\mathcal{JPT}^G$ that on input of length n makes $O(\log(q(n))) = O(n^\alpha)$ calls to G and is a $(q(n), 2^{c'n}, 2^{c'n})$-PRF for every $0 < c' < c$.*

A restriction of Theorem B.2 is that it dictates that the resulting PRF family makes *at least* $\Omega(\sqrt{n})$ calls to the underlying PRG (since $1/2 \leq \alpha < 1$). We note that the restriction that $\alpha > 1/2$ (and hence $q(n) > 2^{\sqrt{n}}$) in the construction of [25] is inherent due to their hashing technique (and is not a mere side-effect of the parameters above).

Using better hashing constructions (based on cuckoo hashing) yields a more versatile version of the above theorem, that in particular allows α to be arbitrary. Specifically, combining Proposition B.1 with Theorem 4.4 yields the following result.

Corollary B.3. *Let G be a length-doubling (t, ε)-PRG. Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$ and $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be efficient $k(n)$-wise independent function family ensembles. Let $q(n) \leq 2^{m(n)-2}$.*

Then, the length-preserving function family ensemble $\{\mathcal{PP}(\mathcal{H}_n, \mathcal{G}_n, \mathcal{GGM}^G_{m(n)\to n})\}_{n\in\mathbb{N}}$, when invoked on input of length n, makes $m(n)$ calls to G and is a $(q, t - p\cdot m\cdot q, 2m\cdot q\cdot\varepsilon + q/2^{\Omega(k)})$-PRF, where $p(\cdot)$ is a polynomial determined by the evaluation and sampling time of $\mathcal{H}$, $\mathcal{G}$ and G.

In particular, for $c > 0$, $0 < \alpha < 1$, $t(n) = 2^{cn}$, $\varepsilon(n) = 1/t(n)$, $q(n) = 2^{n^\alpha}$, $m(n) = \Theta(\log(q(n)))$ and $k(n) = \Theta(n^\alpha + cn)$, the function family $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{GGM}^G_{m(n)\to n})$ makes $m(n) = O(n^\alpha)$ calls to G and is a $(q(n), 2^{c'n}, 2^{c'n})$-PRF, for every $0 < c' < c$.

Proof. We prove the "In particular" part of the corollary. Set $q(n) = 2^{n^\alpha}$, $m(n) = \lceil n^\alpha \rceil + 2$, and $\mathcal{H}$ and $\mathcal{G}$ to be $k(n)$-wise independent for $k(n) = \Theta(n^\alpha + cn)$, with an appropriate constant, such that $\frac{q(n)}{2^{\Omega(k(n))}} < 2^{-cn}$. Let $t' = t - p\cdot m\cdot q$ and $\varepsilon' = 2m\cdot q\cdot\varepsilon + q/2^{\Omega(k)}$. By the first part of the corollary we get that $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{GGM}^G_{m(n)\to n})$ makes $m(n)$ calls to G and is a (q, t', ε')-PRF. Next, we show that $t(n) > 2^{c'n}$ and $\varepsilon(n) < 2^{-c'n}$ for large enough n.

Let $c'' \in \mathbb{N}$ such that $n^{c''} > p(n)$ for large enough n. It follows that $t(n) > 2^{cn} - n^{c''}2^{n^\alpha}(n^\alpha + 2)$ and $\varepsilon(n) < 2^{1+\log(n^\alpha+2)+n^\alpha-cn} + 2^{-cn}$. Hence, for every $c' < c$, we have $\varepsilon(n) < 2^{-c'n}$ and $t(n) > 2^{c'n}$ for large enough n, as required. □

Comparison with the Jain et al. reduction The advantage of Corollary B.3 is that when the adversaries are allowed to make less than $2^{\sqrt{n}}$ queries, the number of calls to the PRG is reduced accordingly, and below $O(\sqrt{n})$ calls. This improves upon the function family $\mathcal{JPT}$, that for such adversaries must make at least $O(\sqrt{n})$ calls to the PRG.

The function family $\mathcal{JPT}$, however, might have shorter description (key) and evaluation time. Specifically, let q denote the number of queries the adversaries are allowed to make (i.e., $q = 2^{n^\alpha}$).

Family	#queries limitation	description (key) size	evaluation time
$\mathcal{JPT}$ [25]	$2^{n^{1/2}} < q < 2^n$	$\Theta(\log q \cdot n)$	$\Theta(\log q \cdot e_G + e_{(\log q)})$
$\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{GGM}_m^G)$	$0 < q < 2^n$	$\Theta(n^2)$	$\Theta(\log q \cdot e_G + e_{(n)})$

Table 2: Comparison between the family $\mathcal{JPT}$ in Theorem B.2 to $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{GGM}_m^G)$ in Corollary B.3. The notation $e_{(k)}$ in the table refers to the evaluation time of a k-wise independent function.

According to Corollary B.3, the parameter k (the independence required) needs to be set to $\Theta(n)$. Hence, by Fact 2.6 it takes $\Theta(n^2)$ bits to describe a function in $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{GGM}_m^G)$. The evaluation time of a single call to $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{GGM}_m^G)$ is $\Theta(\log q \cdot e_G + e_{(n)})$, where e_G is the evaluation time of G and $e_{(k)}$ is the evaluation time of a k-wise independent function. In comparison, it takes $\Theta(\log q \cdot n)$ bits to describe a member in $\mathcal{JPT}$, and its evaluation time is $\Theta(\log q \cdot e_G + e_{(\log q)})$. This is summarize in Table 2.

Independent work. Independently and concurrently with this work, Chandran and Garg [12] showed that a variant of the construction of [25] achieves similar security parameters to [25] and also works for 2^{n^α} queries for any $0 < \alpha < 1/2$. The construction of [12], however, outputs only $n^{2\alpha}$ bits, as opposed to n bits in the construction of [25] and in our construction.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de