Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 182 (2012)

Hardness Preserving Reductions via Cuckoo Hashing

Itay Berman* Iftach Haitner* Ilan Komargodski† Moni Naor†‡

December 24, 2012

Abstract

A common method for increasing the usability and uplifting the security of pseudorandom function families (PRFs) is to "hash" the inputs into a smaller domain before applying the PRF. This approach, known as "Levin's trick", is used to achieve "PRF domain extension" (using a short, e.g., fixed, input length PRF to get a variable-length PRF), and more recently to transform non-adaptive PRFs to adaptive ones. Such reductions, however, are vulnerable to a "birthday attack": after $\sqrt{|\mathcal{U}|}$ queries to the resulting PRF, where $\mathcal{U}$ being the hash function range, a collision (i.e., two distinct inputs have the same hash value) happens with high probability. 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 birthday attack barrier, by replacing the above simple hashing approach with a variant of *cuckoo hashing* — a hashing paradigm typically used for resolving hash collisions in a table, by using two hash functions and two tables, and cleverly assigning each element into 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 non cryptographic work. (ii) A *security-preserving* reduction from non-adaptive to adaptive PRFs.

1 Introduction

We study methods for strengthening pseudorandom functions; such methods transform functions that are somewhat weak (e.g., with a *small* domain, or withstands only non-adaptive (also known as static) attacks: the attacker choose its query ahead of time, before seeing any of the answers) into 'stronger' functions, e.g., with a *large* domain, or secure against *adaptive* (dynamic) attacks: the attacker queries may be chosen as a function of all previous answers.

A common paradigm, first suggested by Levin [23, §5.4], for increasing the usability and uplifting the security of pseudorandom function families (PRFs), is to "hash" the inputs into a smaller domain before applying the PRF. This approach is used to achieve "PRF domain extension" (using

*School of Computer Science, Tel Aviv University. E-mail: {itayberm@post,iftachh@cs}.tau.ac.il. Research supported in part by Check Point Institute for Information Security and the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

†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 of the Planning and Budgeting Committee and the Israel Science Foundation.

‡Incumbent of the Judith Kleeman Professorial Chair. Research supported in part by a grant from the Israel Science Foundation.

ISSN 1433-8092

a short, e.g., fixed, input length PRF to get a variable-length PRF), and more recently to transform non-adaptive PRFs to adaptive ones [6]. Such reductions, however, are vulnerable to the following "birthday attack": after $\sqrt{|\mathcal{U}|}$ queries to the resulting PRF, where $\mathcal{U}$ being the hash function range, a collision (i.e., two distinct inputs have the same hash value) happens with high probability. Such collisions are an obstacle for the indistinguishability of the PRF, since in a random function we either do not expect to see a collision at all (in case 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. Specifically, we consider 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, and assigning each element to one of the two tables (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. A second implication is a *security-preserving reduction* from non-adaptive to adaptive PRFs, an improvement upon the recent result of [6].

Before stating our results, we discuss in more details the notions of pseudorandom functions and cuckoo hashing.

1.1 Pseudorandom Functions

Pseudorandom function families (PRFs), introduced by Goldreich, Goldwasser, and Micali [16], are function families that cannot be distinguished from a family of *truly* random functions, by an efficient distinguisher who is given an oracle access to a random member of the family. PRFs have an extremely important role in cryptography, allowing parties, which share a common secret key, to send secure messages, identify themselves and to authenticate messages [15, 24]. In addition, they have many other applications, essentially in any setting that requires random function provided as black-box [4, 8, 11, 14, 25, 34]. Different PRF constructions are known in the literature, whose security are based on different hardness assumptions. The construction that is most relevant to this work is the one of [16], hereafter the GGM construction, that uses a length-doubling pseudorandom generator (and thus can be based on the existence of one-way functions [18]).

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 *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, in case q is only limited by t and $\varepsilon = 1/t$.

We note that the information-theoretic analog of a t-PRF is known as a t-wise independent family and is formally defined in Definition 2.5.

1.2 Cuckoo Hashing and Many-wise independent hash function

Cuckoo hashing, introduced by Pagh and Rodler [36], 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, and applying a clever algorithm for placing at most a single element in each cell. Since its introduction, many variants of cuckoo hashing have been proposed and an extensive literature is devoted to its analysis.

Pagh and Pagh [35] used ideas in the spirit of cuckoo hashing to construct efficient many-wise independent hash functions. For that they introduce the following paradigm. 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. 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 [35] showed that when the families $\mathcal{H}$ and $\mathcal{G}$ are of "high enough" independence, roughly, both families are $(c\cdot\log|\mathcal{S}|)$-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 being the set of all functions from $\mathcal{S}$ to $\mathcal{R}$.[1] Note that the security of the resulting family goes well beyond the birthday attack barrier: it is indistinguishable from random by an attacker making $|\mathcal{S}| >> \sqrt{|\mathcal{S}|}$ queries.

Aumüller et al. [3] (building on the work of Dietzfelbinger and Woelfel [12]) were able to strengthen the result of Pagh and Pagh [35] by using more sophisticated hash functions $\mathcal{H}$ and $\mathcal{G}$ (rather than the $O(\log|\mathcal{S}|)$-wise independent that [35] used). Specifically, for a given $s\geq 0$, Aumüller et al. [3] constructed a function family $\mathcal{ADW}(\mathcal{H},\mathcal{G},\Pi)$ such that the family $\mathcal{ADW}(\mathcal{H},\mathcal{G},\Pi)$ is $O(|\mathcal{S}|^{-(s+1)})$-indistinguishable from random by a $|\mathcal{S}|$-query , *non-adaptive* distinguisher, where Π being the set of all functions from $\mathcal{S}$ to $\mathcal{R}$, as above. The idea to use more sophisticated hash functions, in the sense that they require less combinatorial work, has 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.[2] 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 $\mathcal{ADW}(\mathcal{H},\mathcal{G},\mathcal{F})$), we get a pseudorandom function that is superior over $\mathcal{F}$ (the actual properties of $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ are determined by the choice 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 [35], Dietzfelbinger and Woelfel [12] and Aumüller et al. [3] went almost unnoticed in the cryptography literature so far.[3] In this work we apply, in a black-box manner, the analysis of [35] and of [3] in cryptographic settings.

[1] We note that in some cases, an adaptive adversary is a more powerful distinguisher than a non-adaptive one. An example of this difference is when an adaptive distinguisher and a non-adaptive distinguisher try to distinguish between a truly random function and a random involution (permutations where the cycle length is at most 2). Clearly, the adaptive distinguisher will succeed with very high probability while the non adaptive distinguisher will fail with very high probability (see [22, 33]).

[2] We note that our approach for this transformation has many predecessors. For instance, the work of Naor and Reingold [32], and of Jetchev et al. [21].

[3] Pagh and Pagh [35] did notice this connection, and in particular mentioned the connection of their work to that of Bellare et al. [5].

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 are used for reducing the cost of a single invocation of the PRF and increasing its usability. Domain extension methods are typically measured by the security of the resulting PRFs, and the amount of calls the resulting PRF makes to the underlying PRF.

A known domain extension technique is the MAC-based constructions, like CBC-MAC and PMAC (a survey on their security can be found in [30]). The number of calls made by these constructions to the underlying (small domain) PRF can be as small as two (for doubling the domain length). Assuming that the underlying PRF is a random function over $\{0,1\}^n$,[4] then the resulting family is $(q,\infty,O(q^2/2^n))$-PRF (i.e., the ∞ in the second parameter means that the distinguisher running time is unlimited). A second technique is using the Feistel or Benes transformations (e.g., [1, 37, 38], where a complete survey on can be found in [39]). The Benes based construction makes 8 calls to the underlying PRF and is $(q,\infty,O(q/2^n))$-PRF, where a 5-round Feistel based construction (that makes 5 calls to the underlying PRF) is $(q,\infty,O(q/2^n))$-PRF. Once you have a length doubling extension one can get any extension at the appropriate cost.

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

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 functions 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 $q \leq 2^{n-2}$.*

Specifically, for $k=\Theta(n)$ Theorem 1.1 yields a domain extension that is $(q,\infty,q/2^n)$-PRF. The resulting PRF makes only two calls to the underlying PRF. Using larger k (i.e. higher independence in terms of combinatorial work) we can decrease the error to be arbitrary small. We note that we actually get stronger result using the function family $\mathcal{ADW}(\mathcal{H},\mathcal{G},\Pi)$ (for different $\mathcal{H}$ and $\mathcal{G}$. See Theorem 4.4).

1.3.2 From PRG to PRF

Another application of the above technique is a hardness preserving construction of PRFs from pseudorandom generators (PRG). An efficiently computable function $G\colon \{0,1\}^n \mapsto \{0,1\}^{2n}$ is t-PRG, if any distinguisher of running time $t(n)$ can distinguish a random output of G from a truly random $2n$-bit string, with probability at most $1/t(n)$. We are interested in constructions of PRF using this PRG that preserve the security of G, more specifically, constructions that are $(2^{c'n})$-PRF for some $0 < c' < c$, assuming that G is a (2^{cn})-PRG.[5] The efficiency of such constructions is

[4] Measuring the security of the resulting under this unrealistic random function assumption is a useful, and common, method for understanding the quality of the domain extension reduction itself.

[5] Considering this range of parameters is only for the sake of concreteness. Our actual result (see Section 4) handles a larger range of parameters.

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. [16] (i.e., GGM) is in fact such hardness preserving according to the above criterion. Their construction, however, makes n calls to the underlying PRG, which might be too expensive is some settings. In order to reduce the number of calls to the underlying PRG, Levin [23] suggested to first hash the input to a smaller domain, and only then apply GGM. The resulting construction, however, is not hardness preserving.

While the GGM construction seems optimal for the security it achieves (as shown in [20]), in some settings the number of queries the distinguisher can make is *strictly less* than its running time; for instance consider a distinguisher of running time 2^{cn} who can only make $2^{\sqrt{n}} \ll 2^{cn}$ queries. In such settings the security of the GGM seems like an overkill, and raises the question whether there exist more efficient reductions. Jain et al. [20] (who raised the above questions) gave the following partial answer, by designing a domain extension method tailored to the PRG to PRF reduction.

Theorem 1.2 ([20], informal)**.** *Let $c > 0$ and $1/2 \leq \alpha < 1$. There exists an efficient oracle-aided function family* JPT *such that the following holds: assume that G is a length-doubling (2^{cn})-PRG, then JPT^G is a $(2^{n^\alpha}, 2^{c'n}, 2^{c'n})$-PRF, for every $0 < c' < c$. A function $f \in \mathsf{JPT}^G$ makes $O(n^\alpha)$ queries to G.*

A restriction of Theorem 1.2 is that it dictates the resulting PRF family to use *at least* $\Omega(\sqrt{n})$ queries to the underlying PRG (since $1/2 \leq \alpha < 1$). Here we use cuckoo hashing to give a more versatile version of their theorem, allowing α to be arbitrary, that also improves some of their parameters. Thus, our result implies hardness-preserving PRF reduction, which is useful to construct PRFs of low query complexity.

In the following we let GGM_m to be the variant of the GGM construction that on input of length $m(n)$, makes $m(n)$ calls to a length-doubling PRG on inputs of length n, and outputs a string of length n.[6]

Corollary 1.3 (informal)**.** *Let $c > 0$ and $0 < \alpha < 1$, let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$ and $\mathcal{G} = \{\mathcal{G} \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be efficient $\Theta(n^\alpha + cn)$-wise independent function families. Assume that G is a length-doubling (2^{cn})-PRG, then $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathsf{GGM}_m^G)$ is a $(2^{\Omega(n^\alpha)}, 2^{c'n}, 2^{c'n})$-PRF, for every $0 < c' < c$, and $m(n) = O(n^\alpha)$.*

Note that $f \in \mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathsf{GGM}_m^G)$ makes $O(n^\alpha)$ queries to G. Again, using $\mathcal{ADW}(\mathcal{H}, \mathcal{G}, \mathsf{GGM}_m^G)$ (with different $\mathcal{H}$ and $\mathcal{G}$), we actually get stronger result (see Corollary 4.9). We refer to Section 4.1.1 for a more elaborate comparison between the construction of [20] and our construction.

Independently and concurrently with this work, Chandran and Garg [10] showed that a variant of the construction of [20] achieves similar security parameters to [20] and also works for 2^{n^α} queries for any $0 < \alpha < 1/2$. The construction of [10], however, outputs only $n^{2\alpha}$ bits, as opposed to n bits as in the construction of [20] and as in our construction.

1.3.3 From Non-Adaptive to Adaptive PRF

Constructing adaptive PRFs from non-adaptive ones can be done using general techniques; for instance, using the PRG-based construction of Goldreich et al. [16] or the *synthesizers* based construction of Naor and Reingold [31]. These constructions, however, make (roughly) n calls to the

[6] GGM_m is a variant of the standard GGM function family, that on input of length $m(n)$ uses seed of length n for the underlying generator, rather than seed of length $m(n)$ (see Proposition 4.6 for the formal definition).

underlying non-adaptive PRF (where n is the input length). Recently, Berman and Haitner [6] showed how to perform this security uplifting at a much lower price: the adaptive PRF makes only a *single* call to the non-adaptive PRF. The drawback of their construction, however, is 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 significant degradation in the security 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 PRF.

Theorem 1.4 (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 $t(n)$-PRF. Then $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ is a length-preserving $(t(n)/4)$-PRF.*

As before, we actually get stronger result using $\mathcal{ADW}(\mathcal{H},\mathcal{G},\mathcal{F})$ (with different $\mathcal{H}$ and $\mathcal{G}$. See Theorem 5.5).

1.4 More Related Work

Bellare et al. [5] introduced a paradigm for using PRFs in the symmetric-key settings that, in retrospect, is similar to cuckoo hashing. Assume 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 in case 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 large domain. Instead, [5] 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 a much better security then 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 [5].

The issue 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 pseudorandom permutations (or block ciphers). Maurer and Pietrzak [27] showed that starting from a family of permutations that is information-theoretic secure against non-adaptive attacks,[7] if two independently chosen members of the family are composed, then the result is a permutation secure against adaptive attacks (see [27] for the exact formulation). On the other hand, Pietrzak [40] showed that this is not necessarily the case for permutations that are randomly looking under a computational assumption (see also [29, 41]) (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. Section 3 is where we formally define the hashing paradigm of [35] and of [3], and show how to extend their results to hold against adaptive

[7] That is, secure against (non-adaptive) unbounded attackers.

adversaries. Our domain extension is described in Section 4, and the improved non-adaptive to adaptive reduction is described in Section 5. In Section 6 we survey some possible directions for future research.

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 $\mathcal{S}^{\leq t} = \{\overline{s} \in \mathcal{S}^* \colon |\overline{s}| \leq t \quad \wedge \quad \overline{s}[i] \neq \overline{s}[j] \quad \forall 1 \leq i < j \leq |\overline{s}|\}$, 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 integers n and ℓ, let $\Pi_{n,\ell}$ stands for the set of all functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, and let $\Pi_n = \Pi_{n,n}$.

We let poly denote the set all polynomials, and let PPTM denote the set of probabilistic algorithms (i.e., Turing machines) that run in *strictly* polynomial time. 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 $(t(n), \varepsilon(n))$-*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.

2.3 Function Families

2.3.1 Operating on Function Families

We consider two natural operation 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 (XOR 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 $\mathcal{R}^1, \mathcal{R}^2 \subseteq \{0,1\}^\ell$. The* XOR *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 $(f_2, f_1)(x) := f_2(x) \oplus f_1(x)$.*

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 contained in $\mathcal{R}_n$. 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 function family ensemble to be useful, it has to have an efficient sampling and evaluation algorithms.

Definition 2.4 (efficient function family ensembles)**.** *A function family ensemble $\mathcal{F} = \{\mathcal{F}_n : \mathcal{D}_n \mapsto \mathcal{R}_n\}_{n\in\mathbb{N}}$ 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 $\mathcal{H} = \{h : \mathcal{D} \mapsto \mathcal{R}\}$ 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 $\ell, k \in \mathrm{poly}$, the existence of efficient $k(n)$-wise independent family ensembles mapping strings of length $\ell(n)$ to strings of length n is well known ([9, 44]). 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. Starting with Siegel [43], there has been quite a lot of attention devoted to the question of whether it is possible to come up with constructions that require much less than k operations per evaluation (see Section 1.2).

As a side remark 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[8] are only granted to be resistant against *non-adaptive* distinguishers. 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 2.6 (Pseudorandom Functions)**.** *An efficient function family ensemble $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ is an (adaptive) $(q(n), t(n), \varepsilon(n))$-*PRF*, if*

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

[8] 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 $\varepsilon > 0$.

for every $q(n)$-query oracle-aided algorithm (distinguisher) D of running time $t(n)$ and large enough n. If $q(n)$ is only bounded by $t(n)$, then $\mathcal{F}$ is called $(t(n),\varepsilon(n))$-PRF. In addition, if we limit D above 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 $(t(n),\varepsilon(n))$-PRF. *Finally, The ensemble $\mathcal{F}$ is a t-PRF, if it is a $(t,1/t)$-PRF according to the above definition (where the same conventions are also used for non-adaptive PRFs).*

Some applications require the pseudorandom functions to be secure against distinguisher with access to two oracles (and not just a single oracle as in Definition 2.6).

Definition 2.7 (pseudorandom functions secure against two-oracle distinguishers)**.** *An efficient function family ensemble $\mathcal{F}=\{\mathcal{F}_n\colon\{0,1\}^n\mapsto\{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ is a $(q(n),t(n),\varepsilon(n))$-*two-oracle PRF*, if for every $q(n)$-query two oracle-aided (i.e., makes at most $q(n)$ queries to each oracle) algorithm (distinguisher) D of running time $t(n)$ and large enough n, it holds that*

$$\left|\Pr_{(f_1,f_2)\leftarrow\mathcal{F}_n\times\mathcal{F}_n}[\mathsf{D}^{f_1,f_2}(1^n)=1]-\Pr_{(\pi_1,\pi_2)\leftarrow\Pi_{n,\ell(n)}\times\Pi_{n,\ell(n)}}[\mathsf{D}^{\pi_1,\pi_2}(1^n)=1]\right|\leq\varepsilon(n).$$

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

Lemma 2.8 (cf., [7], Theorem 1)**.** *Let $\mathcal{F}=\{\mathcal{F}_n\colon\{0,1\}^n\mapsto\{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be a function family ensemble. Then for every $q(n)$-query* two-oracle *adaptive [resp., non-adaptive] distinguisher D of running time $t(n)$, there exists a $q(n)$-query* single-oracle *adaptive [resp., non-adaptive] distinguisher $\widehat{\mathsf{D}}$ of running time $t(n)+2q(n)e_{\mathcal{F}}(n)$, where $e_{\mathcal{F}}$ stands for the evaluation time of $\mathcal{F}$, with*

$$\begin{aligned}&\left|\Pr_{f\leftarrow\mathcal{F}_n}[\widehat{\mathsf{D}}^{f}(1^n)=1]-\Pr_{\pi\leftarrow\Pi_{n,\ell(n)}}[\widehat{\mathsf{D}}^{\pi}(1^n)=1]\right|\\&\geq\frac{1}{2}\cdot\left|\Pr_{(f_1,f_2)\leftarrow\mathcal{F}_n\times\mathcal{F}_n}[\mathsf{D}^{f_1,f_2}(1^n)=1]-\Pr_{(\pi_1,\pi_2)\leftarrow\Pi_{n,\ell(n)}\times\Pi_{n,\ell(n)}}[\mathsf{D}^{\pi_1,\pi_2}(1^n)=1]\right|\end{aligned}$$

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

Lemma 2.8 is proven using a standard hybrid argument. For a complete proof we refer the reader to [7], where a more general lemma is proven.[9]

3 From Non-Adaptive to Adaptive Hashing

In this section we describe a general transformation of non-adaptive secure function families with a certain combinatorial property into adaptive secure ones. We note that the transformations defined in this section *cannot* be applied directly to (non-adaptive) PRFs, since we have no reason to assume that such families posses this property (alternatively, see Section 5 for the transformation from non-adaptive to adaptive PRFs).

Our framework can be used to prove that for certain function families, adaptive distinguishers are subject to the same distinguishing bound of non-adaptive ones. Specifically, it deals with constructions where the randomness can be partitioned into two (non empty) parts $\mathcal{U}$ and $\mathcal{V}$ and there exists some bad event that is defined only over the $\mathcal{U}$ part for a given subset of the domain (the queries). In addition, if the bad event does not happen, we require that the resulting output will be uniform over the subset of queries. We begin with a definition of *monotone sets*.[10]

[9] [7] only states, and proves, the adaptive case, but the very same lines also yields the non-adaptive case.

[10] Monotone sets are a special case of the "monotone event sequence" defined in [26].

Definition 3.1 (monotone 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 $(\overline{s}_2, t) \in \mathcal{M}$.*

Next, we formally state the lemma that is the basis of our framework. The lemma deals with a construction of a function family $\mathcal{F}$ that can be defined as $\mathcal{F} = \mathcal{F}(\mathcal{U}, \mathcal{V})$ where $\mathcal{U}$ and $\mathcal{V}$ are arbitrary non-empty sets. Intuitively, it states that assuming there exists a bad event BAD that can be defined over the inputs and the $\mathcal{U}$ part (i.e, independently of the $\mathcal{V}$ part) that happens with small probability, and conditioning on that BAD does not happen, we know that $\mathcal{F}$ is uniform over subsets of the range of size that is the number of queries, then we can say that the function family $\mathcal{F}$ is resistant against adaptive adversaries.

Lemma 3.2. *Let $\mathcal{F} = \mathcal{F}(\mathcal{U}, \mathcal{V})$ be a function family of the form $\{f_{u,v} \colon \mathcal{D} \mapsto \mathcal{R}\}_{(u,v) \in \mathcal{U} \times \mathcal{V}}$, where $\mathcal{U}$ and $\mathcal{V}$ are arbitrary non-empty sets, let $t \in \mathbb{N}$ and let $\mathrm{BAD} \subseteq \mathcal{D}^{\leq t} \times \mathcal{U}$ be a left-monotone set. Assume that for every $\overline{q} \in \mathcal{D}^{\leq t}$ it holds that*

1. *$\left(f(\overline{q}_1), \ldots, f(\overline{q}_{|\overline{q}|})\right)_{f \leftarrow \{f_{u,v} \colon v \in \mathcal{V}\}}$ is uniform over $\mathcal{R}^{|\overline{q}|}$ for every $u \in \mathcal{U}$ such that $(\overline{q}, u) \notin \mathrm{BAD}$, and*
2. *$\Pr_{u \leftarrow \mathcal{U}}[(\overline{q}, u) \in \mathrm{BAD}] \leq \varepsilon$,*

then for any t-query adaptive *algorithm* D*, it holds that*

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

where Π is the set of all functions from $\mathcal{D}$ to $\mathcal{R}$.

Lemma 3.2 is a special case of a result given in [21, Theorem 12] (which closes a gap in the work of [26]), and its direct proof can be found in Appendix A.

3.1 Instantiation with the Pagh and Pagh [35] Function Family

In this section we instantiate the framework with the function family of Pagh and Pagh [35]. One advantage of this construction is the relative simplicity of description. We begin by describing their method of combining function families.

Definition 3.3 (The Pagh and Pagh [35] function family)**.** *Let $\mathcal{H}$ be a function family from $\mathcal{D}$ to $\mathcal{S}$, 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 $\mathcal{R}$. Define the function family $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F})$ from $\mathcal{D}$ to $\mathcal{R}$ as*

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

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

Pagh and Pagh [35] showed that when instantiated with the proper function families, the above function family has the following properties:

Theorem 3.4 ([35])**.** *Let t be an integer, let $\mathcal{H} = \{h \colon \mathcal{D} \mapsto [4t]_{\mathcal{D}}\}$ and $\mathcal{G} = \{g \colon \mathcal{D} \mapsto \mathcal{R}\}$ be function families, and let Π be the all function family from $\mathcal{D}$ to $\mathcal{R}$. Then for every $k, t \in \mathbb{N}$ there exists left-monotone set $\mathrm{BAD} \subseteq \mathcal{D}^{\leq t} \times \mathcal{H}^2$,[11] such that the following holds for every $\overline{q} \in \mathcal{D}^{\leq t}$:*

[11] See [35] for the definition of BAD.

1. *Assuming that $\mathcal{G}$ is k-wise independent over the elements of $\overline{q}$, then* $\left(f(\overline{q}_1),\ldots,f(\overline{q}_{|\overline{q}|})\right)_{f\leftarrow\mathcal{PP}_{h_1,h_2}(\mathcal{G},\Pi)}$ *is uniform over $\mathcal{R}^{|\overline{q}|}$ for every $u\in\mathcal{U}$ such that $(\overline{q},u)\notin$ BAD.*

2. *Assuming that $\mathcal{H}$ is k-wise independent over the elements of $\overline{q}$, then* $\Pr_{u\leftarrow\mathcal{H}^2}[(\overline{q},u)\in\text{BAD}]\leq t/2^{\Omega(k)}$.[12]

What Pagh and Pagh [35] concluded is that for (the many) applications where the analysis is applied to a static set it is safe to use this family. However, as we can see, the function family $\mathcal{PP}(\mathcal{H},\mathcal{G},\Pi)$ is not only closed 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 $\mathcal{H}$, $\mathcal{G}$ and Π be as in Theorem 3.4, and let* D *be an* adaptive*, t-query oracle-aided algorithm. 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 $\mathcal{U}=\mathcal{H}\times\mathcal{H}$, $\mathcal{V}=\Pi\times\Pi\times\mathcal{G}$. 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 assures that the first property of Lemma 3.2 is satisfied, and according to Item 2 of Theorem 3.4 we set ε of Lemma 3.2 to be $t/2^{\Omega(k)}$, and thus the second property is also satisfied. Hence, applying Lemma 3.2 with respect to the above sets, concludes the proof of the lemma. □

Remark 3.6. *For our application, see Sections 4 and 5, 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., [6]) 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.*

3.2 Instantiation with the Aumüller et al. [3] Function Family

We now explore instantiating the framework with the function families of Aumüller et al. [3]. The resulting families enjoy shorter description length and invoking them require less combinatorial work than [35] based families discussed above. On the other hand, describing them on paper is a bit more complicated. The function family of Aumüller et al. [3] (building upon Dietzfelbinger and Woelfel [12]) follows the same basic outline as the [35] function family, but uses more complex hash functions. Recall that the members of the Pagh and Pagh [35] function family $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ are of

[12] The function family we consider above (i.e., $\mathcal{PP}$) is slightly different than the one given in [35]. 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 [35] original proof (specifically from Lemma 3.3 and 3.4).

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 family $\mathcal{ADW}(\mathcal{H}, \mathcal{G}, \mathcal{F})$ described below, the role of h_1, h_2 and g is taken by some variant of *tabulation hashing* (and not taken from a relatively high k-wise independent family as in [35]). Specifically, at the heart of these functions lies a function of the form:

$$a_{h,\overline{g},M}(x) := \left(h(x) + \sum_{1 \le j \le z} M[\overline{g}_j(x), j] \right) \bmod m$$

where $M \in (\mathbb{Z}_m)_{\ell \times z}$ is a $\ell \times z$ matrix, $\overline{g} = (g_1, \cdots, g_z)$ is a list of z functions where $g_j \colon \mathcal{D} \mapsto \mathbb{Z}_\ell$ and $h \colon \mathcal{D} \mapsto \mathbb{Z}_m$. The matrix M will be chosen at random (sometimes actually pseudorandomly) and the g_j's and h from a relatively low independence family. In addition, unlike in Pagh and Pagh [35], the functions are chosen in a *correlated* manner (i.e., , sharing the *same* function vector $\overline{g}$).

In the rest of this section, we formally define the hash function family of Aumüller et al. [3], state their (non-adaptive) result, and apply Lemma 3.2 to get an adaptive variant of their result.

Definition 3.7 (The Aumüller et al. [3] function family)**.** *For $z \in \mathbb{N}$, for functions $h_1, h_2, h_3 \colon \mathcal{D} \mapsto \mathbb{Z}_m$ and $f_1, f_2 \colon \mathcal{D} \mapsto \mathcal{R}$, a function vector $\overline{g} = (g_1, \cdots, g_z)$, where $g_j \colon \mathcal{D} \mapsto \mathbb{Z}_\ell$ for each $1 \le j \le z$, and matrices $M_1, M_2, M_3 \in (\mathbb{Z}_m)_{\ell \times z}$, define the function $\mathsf{adw}_{M_1,M_2,M_3,h_1,h_2,h_3,\overline{g},f_1,f_2}$ from $\mathcal{D}$ to $\mathcal{R}$ as*

$$\mathsf{adw}_{M_1,M_2,M_3,h_1,h_2,h_3,\overline{g},f_1,f_2} := (f_1 \circ a^{\mathcal{D}}_{h_1,\overline{g},M_1}) \oplus (f_2 \circ a^{\mathcal{D}}_{h_2,\overline{g},M_2}) \oplus a^{\mathcal{R}}_{h_3,\overline{g},M_3}, \tag{2}$$

where $a^{\mathcal{S}}_{h,\overline{g},M}(x)$, for a set $\mathcal{S}$, is the $(a_{h,\overline{g},M}(x))^{th}$ element of $\mathcal{S}$ (in lexicographic order), where $a_{h,\overline{g},M}(x) := \left(h(x) + \sum_{1 \le j \le z} M[\overline{g}_j(x), j] \right) \bmod m$.

For function families $\mathcal{H} = \{h \colon \mathcal{D} \mapsto \mathbb{Z}_m\}$ and $\mathcal{F} = \{f \colon \mathcal{D} \mapsto \mathcal{R}\}$, and $M_1, M_2, h_1, h_2, \overline{g}$ as above, let

$$\mathcal{ADW}_{M_1,M_2,h_1,h_2,\overline{g}}(\mathcal{H}, \mathcal{F}) := \{\mathsf{adw}_{M_1,M_2,M_3,h_1,h_2,h_3,\overline{g},f_1,f_2} \colon M_3 \in (\mathbb{Z}_m)_{\ell \times z}, h_3 \in \mathcal{H}, f_1, f_2 \in \mathcal{F}\} \tag{3}$$

Finally, for function family $\mathcal{G} = \{g \colon \mathcal{D} \mapsto \mathbb{Z}_\ell\}$, and $\mathcal{H}, \mathcal{F}$ as above, let

$$\begin{aligned} \mathcal{ADW}_z(\mathcal{H}, \mathcal{G}, \mathcal{F}) := \{&\mathsf{adw}_{M_1,M_2,M_3,h_1,h_2,h_3,\overline{g},f_1,f_2} \colon \\ &M_1, M_2, M_3 \in (\mathbb{Z}_m)_{\ell \times z}, h_1, h_2, h_3 \in \mathcal{H}, \overline{g} \in \mathcal{G}^z, f_1, f_2 \in \mathcal{F}\}. \end{aligned} \tag{4}$$

Aumüller et al. [3] proved the following result with respect to the above function family.

Theorem 3.8 ([3])**.** *The following holds for any $t, s \in \mathbb{N}$ and $\zeta, c_1, c_2 > 0$: let $m = (1+\zeta)t$, $\delta = c_1 / \log t$, $\ell = t^\delta$, $k = c_2 \cdot s \cdot \log t$ and $z = \lceil (s+2)/(c_1 \cdot c_2 \cdot s) \rceil$. Let $\mathcal{H} = \{h \colon \mathcal{D} \mapsto \mathbb{Z}_m\}$ and $\mathcal{G} : \{g : \mathcal{D} \mapsto \mathbb{Z}_\ell\}$ be $2k$-wise independent hash families, and let Π be the all function family from $\mathcal{D}$ to $\mathcal{R}$.*

Then for $t \in \mathbb{N}$ there exists a left-monotone set $\mathrm{BAD} \subseteq \mathcal{D}^{\le t} \times \left((\mathbb{Z}_m)^2_{\ell \times z} \times \mathcal{H}^2 \times \mathcal{G}^z \right)$,[13] such that the following holds for every $\overline{q} \in \mathcal{D}^{\le t}$:

1. *$\left(f(\overline{q}_1), \ldots, f(\overline{q}_{|\overline{q}|}) \right)_{f \leftarrow \mathcal{ADW}_u(\mathcal{H},\Pi)}$ is uniform over $\mathcal{R}^{|\overline{q}|}$ for every $u \in (\mathbb{Z}_m)^2_{\ell \times z} \times \mathcal{H}^2 \times \mathcal{G}^z$ such that $(\overline{q}, u) \notin \mathrm{BAD}$, and*

[13] See [3] for the definition of BAD.

2. $\Pr_{u \leftarrow (\mathbb{Z}_m)^2_{\ell \times z} \times \mathcal{H}^2 \times \mathcal{G}^z}[(\overline{q}, u) \in \mathrm{BAD}] \leq 1/t^{s+1}$.

That is, for the right choice of parameters, the function family $\mathcal{ADW}_z(\mathcal{H}, \mathcal{G}, \mathcal{F})$ is not only closed 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. Indeed, by plugging in Theorem 3.8 into the general framework lemma (Lemma 3.2), we get the following result:

Lemma 3.9. *Let s, z, $\mathcal{H}$, $\mathcal{G}$ and Π be as in Theorem 3.8 and let D be an* adaptive*, t-query oracle-aided algorithm. Then*

$$\left|\Pr_{f \leftarrow \mathcal{ADW}_z(\mathcal{H},\mathcal{G},\Pi)}[\mathsf{D}^f = 1] - \Pr_{\pi \leftarrow \Pi}[\mathsf{D}^\pi = 1]\right| \leq 1/t^{s+1}.$$

Proof. Let $\mathcal{U} = (\mathbb{Z}_m)_{\ell \times z} \times (\mathbb{Z}_m)_{\ell \times z} \times \mathcal{H} \times \mathcal{H} \times \mathcal{G}^z$, $\mathcal{V} = (\mathbb{Z}_m)_{\ell \times z} \times \Pi \times \Pi \times \mathcal{H}$. For $(M_1, M_2, h_1, h_2, \overline{g}) \in \mathcal{U}$ and $(M_3, \pi_1, \pi_2, h) \in \mathcal{V}$, let $F_{(M_1,M_2,h_1,h_2,\overline{g}),(M_3,\pi_1,\pi_2,h)} = (\pi_1 \circ a_{h_1,\overline{g},M_1}) \oplus (\pi_2 \circ a_{h_2,\overline{g},M_2}) \oplus a_{h,\overline{g},M_3}$, 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 3.8. We prove the lemma showing that the above sets meet the requirements stated in Lemma 3.2.

Item 1 of Theorem 3.8 assures that the first property of Lemma 3.2 is satisfied, and according to Item 2 of Theorem 3.8 we set ε of Lemma 3.2 to be $1/t^{s+1}$, 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 t, in contrast to the function family using $\mathcal{PP}$, using $\mathcal{ADW}$ we get meaningful results even when using an underlying $k = O(\log t)$-wise independent family.

Remark 3.10. *In the rest of the paper we mostly apply Lemma 3.9 with the parameters stated in Theorem 3.8. Different choice of parameters can be used to get different results. For example, using larger "random tables" (e.g., $\delta = 1/2$), we improve the result of Jain et al. [20] (see Section 4.1).*

4 Extending the Domain of a PRF

In this section we show how to apply the constructions of Section 3, inspired by cuckoo hashing, in order to extend a domain of a given PRF $\mathcal{F}$ to an arbitrary size one.

Let $\mathcal{P}(\mathcal{U}, \mathcal{V}) = \{\mathcal{P}(\mathcal{U}_n, \mathcal{V}_n)\}_{n \in \mathbb{N}}$ be the ensemble of function families $\{P_{u,v} \colon \mathcal{D}_n \mapsto \mathcal{R}_n\}_{(u,v) \in \mathcal{U}_n \times \mathcal{V}_n}$, where $\mathcal{U}_n$ and $\mathcal{V}_n$ are some non-empty sets as described in Section 3. We begin by showing that $\mathcal{P}(\mathcal{U}, \mathcal{V}^{\mathcal{F}})$ is computationally indistinguishable from $\mathcal{P}(\mathcal{U}, \mathcal{V}^{\mathsf{\Pi}})$, where $\mathcal{V}^{\mathcal{F}}$ denotes that $\mathcal{V}$ is implemented using pseudorandom functions and $\mathcal{V}^{\mathsf{\Pi}}$ denotes that $\mathcal{V}$ is implemented using truly random functions.[14] In the sequel, we assume that $\mathcal{V}$ is implemented using 2 calls to the underlying functions (as this is the case in the actual constructions we work with).

Lemma 4.1. *Let $\mathcal{P}(\mathcal{U}, \mathcal{V}) = \{\mathcal{P}(\mathcal{U}_n, \mathcal{V}_n)\}_{n \in \mathbb{N}}$ be the ensemble of function families $\{P_{u,v} \colon \mathcal{D}_n \mapsto \mathcal{R}_n\}_{(u,v) \in \mathcal{U}_n \times \mathcal{V}_n}$, where $\mathcal{U}_n$ and $\mathcal{V}_n$ are arbitrary sets, $\mathsf{\Pi} = \{\Pi_n \colon \mathcal{S}_n \mapsto \mathcal{R}_n\}_{n \in \mathbb{N}}$, where Π_n is the set of all functions from $\mathcal{S}_n$ to $\mathcal{R}_n$, and $\mathcal{F} = \{\mathcal{F}_n \colon \mathcal{S}_n \mapsto \mathcal{R}_n\}_{n \in \mathbb{N}}$ be an efficient function family. Then*

[14] Namely, in case the implementation of $\mathcal{PP}$ is used, then $\mathcal{U} = \mathcal{H} \times \mathcal{H}$, $\mathcal{V}^{\mathcal{F}} = \mathcal{F} \times \mathcal{F} \times \mathcal{G}$ and $\mathcal{V}^{\mathsf{\Pi}} = \Pi \times \Pi \times \mathcal{G}$. In case the implementation of $\mathcal{ADW}_z$ is used, then $\mathcal{U} = (\mathbb{Z}_m)_{\ell \times z} \times (\mathbb{Z}_m)_{\ell \times z} \times \mathcal{H} \times \mathcal{H} \times \mathcal{G}^z$, $\mathcal{V}^{\mathcal{F}} = (\mathbb{Z}_m)_{\ell \times z} \times \mathcal{F} \times \mathcal{F} \times \mathcal{H}$ and $\mathcal{V}^{\mathsf{\Pi}} = (\mathbb{Z}_m)_{\ell \times z} \times \Pi \times \Pi \times \mathcal{H}$.

for every $q(n)$-query oracle-aided distinguisher D of running time $t(n)$, there exists $p \in \text{poly}$ and a $q(n)$-query distinguisher $\widehat{\mathsf{D}}$ of running time $t(n)+p(n)q(n)$, with

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

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

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

Algorithm 4.2 ($\widetilde{\mathsf{D}}$)**.**

Input: 1^n.

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

1. *Set $f=\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\phi_1,\phi_2})$*
2. *Emulate $\mathsf{D}^f(1^n)$.*

Note that $\widetilde{\mathsf{D}}$ makes $q(n)$ queries to ϕ_1 and ϕ_2, and it can be implemented to run in time $t(n)+q(n)e_{\mathcal{P}}(n)$, where $e_{\mathcal{P}}$ is the sampling and evaluation time of $\mathcal{P}(\mathcal{U},\mathcal{V})$. Observe that in case ϕ_1 and ϕ_2 are uniformly drawn from $\mathcal{F}_n$, then the emulation of D done in $\widetilde{\mathsf{D}}^{\phi_1,\phi_2}$ is identical to a random execution of D^f with $f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\mathcal{F}_n})$. Similarly, in case ϕ_1 and ϕ_2 are uniformly drawn from Π_n, then the emulation is identical to a random execution of D^f with $f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\Pi_n})$. Thus,

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

Hence, Lemma 2.8 yields that there exists a $q(n)$-query single-oracle distinguisher $\widehat{\mathsf{D}}$ of running time $t(n)+q(n)e_{\mathcal{P}}(n)\leq t(n)+p(n)q(n)$, where $p\in\text{poly}$, such that

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

□

Assuming that $\mathcal{D}_n=\{0,1\}^{\ell(n)}$, $\mathcal{R}_n=\{0,1\}^{s(n)}$, $\mathcal{F}=\{\mathcal{F}_n\colon\{0,1\}^{m(n)}\mapsto\{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ is $(q(n),t(n),\varepsilon(n))$-PRF and $\Pi=\{\Pi_{m(n),s(n)}\}_{n\in\mathbb{N}}$, Lemma 4.1 yields that for large enough n

$$\left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\mathcal{F}_n})}[\mathsf{D}^f(1^n)=1]-\Pr_{f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\Pi_{m(n),s(n)}})}[\mathsf{D}^f(1^n)=1]\right|\leq 2\varepsilon(n). \tag{5}$$

Assuming the conditions of Lemma 3.2 with respect to $\mathrm{P}(\mathcal{U},\mathcal{V})$ hold, we get that for some ε'

$$\left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\Pi_{m(n),s(n)}})}[\mathsf{D}^f(1^n)=1]-\Pr_{\pi\leftarrow\Pi_{\ell(n),s(n)}}[\mathsf{D}^\pi(1^n)=1]\right|\leq\varepsilon'(n) \tag{6}$$

for every $n \in \mathbb{N}$. Hence, by the triangle inequality

$$\left|\Pr_{f \leftarrow \mathcal{P}(\mathcal{U}_n, \mathcal{V}_n^{\mathcal{F}_n})}[\mathsf{D}^f(1^n) = 1] - \Pr_{\pi \leftarrow \Pi_{\ell(n),s(n)}}[\mathsf{D}^\pi(1^n) = 1]\right| \leq 2\varepsilon(n) + \varepsilon'(n)$$

for large enough n.

Plugging in a specific function family $\mathcal{P}$ (e.g., $\mathcal{PP}$ or $\mathcal{ADW}$), we get domain extension for a PRF. Specifically, plugging is Lemma 3.5 we get the following theorem.

Theorem 4.3 (Restating Theorem 1.1)**.** *Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$ and $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ be efficient $k(n)$-wise independent function family ensembles, and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{m(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ be a $(q(n), t(n), \varepsilon(n))$-PRF. Then $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathcal{F}) = \{\mathcal{PP}(\mathcal{H}_n, \mathcal{G}_n, \mathcal{F}_n) \colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ is a $(q(n), t(n) - p(n)q(n), 2\varepsilon(n) + q(n)/2^{\Omega(k(n))})$-PRF, where $p \in$ poly is determined by the evaluation and sampling time of $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{F}$ and $q(n) \leq 2^{m(n)-2}$.*[15]

Notice that in order for Theorem 4.3 to be useful, we have to set $k(n) = \Omega(\log q(n))$. Plugging in Lemma 3.9 we get the following theorem.

Theorem 4.4. *Let $s \geq 0$, z, $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$, $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be as defined in Theorem 3.8, and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{m(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ be a $(q(n), t(n), \varepsilon(n))$-PRF. Then $\mathcal{ADW}_z(\mathcal{H}, \mathcal{G}, \mathcal{F}) = \{\mathcal{ADW}_z(\mathcal{H}_n, \mathcal{G}_n, \mathcal{F}_n) \colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^{s(n)}\}_{n\in\mathbb{N}}$ is a $(q(n), t(n) - p(n)q(n), 2\varepsilon(n) + 1/q(n)^{s+1})$-PRF, where $p \in$ poly is determined by the evaluation and sampling time of $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{F}$ and $q(n) \leq 2^{m(n)}/(1+\zeta)$ where ζ is from Theorem 3.8.*

Notice that we used the setting of parameters of Theorem 3.8. In particular, since the matrices that are sampled in the definition of $\mathcal{ADW}_z$ are of constant size, they can be embedded in the key of the PRF.

4.1 Hardness Preserving PRG to PRF Reductions

An important corollary of Theorem 4.3 is a security preserving reduction from pseudorandom generators to pseudorandom functions.

Definition 4.5 (PRG to PRF reductions)**.** *An oracle-aided function family ensemble $\mathcal{F}$ is a (v, q, t, ε)-*PRG-to-PRF *reduction, if the following holds:*

1. *For any oracle G and $n \in \mathbb{N}$, a function $f \in \mathcal{F}_n^G$ makes at most $v(n)$ oracle calls per invocation.*
2. *Assuming that G is a length-doubling (t_G, ε_G)-PRG of evaluation time e_G, then $\mathcal{F}^G$ is a $(q(t_G, \varepsilon_G, e_G), t(t_G, \varepsilon_G, e_G), \varepsilon(t_G, \varepsilon_G, e_G))$-PRF.*

The following fact easily follows from [16].

Proposition 4.6 ([16])**.** *For any integer function m, there exists an efficient oracle-aided function family ensemble, denoted GGM_m, that maps strings of length $m(n)$ to strings of length n, and is a $(m(n), q(n), t_G(n) - m(n) \cdot q(n) \cdot e_G(n), m(n) \cdot q(n) \cdot \varepsilon_G(n))$-PRG-to-PRF reduction for any integer function q.*[16]

[15] The -2 factor is due to the definition of $\mathcal{H}$ is Theorem 3.4.

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

Combining Proposition 4.6 with Theorem 4.3 yields the following result.

Corollary 4.7. *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, then the oracle-aided function ensemble $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathsf{GGM}_m^G)$ is a $(m(n), q(n), t_G(n) - p(n) \cdot m(n) \cdot q(n), 2m(n) \cdot q(n) \cdot \varepsilon_G(n) + q(n)/2^{\Omega(k(n))})$-PRG-to-PRF reduction, where $p \in$ poly is determined by the evaluation and sampling time of $\mathcal{H}$, $\mathcal{G}$ and G, and $q(n) \leq 2^{m(n)-2}$.*

For settings of interest, Corollary 4.7 yields the following result.

Corollary 4.8 (Restating Corollary 1.3)**.** *Let $c > 0$, $0 < \delta < 1$ and $0 < \alpha < \delta$, and let $\mathcal{H}$ and $\mathcal{G}$ be as in Corollary 4.7, with respect to $k(n) = \Theta(n^\alpha + cn^\delta)$ (the hidden constant is universal). Then $\mathcal{PP}(\mathcal{H}, \mathcal{G}, \mathsf{GGM}_m^G)$ is an $(O(n^\alpha), 2^{n^\alpha}, 2^{c'n^\delta}, 2^{-c'n^\delta})$-PRG-to-PRF reduction, for every $0 < c' < c$.*

Proof. Let $t(n) = t_G(n) - p(n) \cdot m(n) \cdot q(n)$ and $\varepsilon(n) = 2m(n) \cdot q(n) \cdot \varepsilon_G(n) + q(n)/2^{\Omega(k(n))}$. Set $k(n) = \Theta(n^\alpha + cn^\delta)$, with an appropriate constant, such that $\frac{q}{2^{\Omega(k)}} < 2^{-cn^\delta}$, and thus $\varepsilon(n) < 2^{1+\log(n^\alpha+2)+n^\alpha-cn^\delta} + 2^{-cn^\delta}$. Let $c'' \in \mathbb{N}$ such that $n^{c''} > p(n)$ for large enough n (where p is of Corollary 4.7), and thus it holds that $t(n) > 2^{cn^\delta} - n^{c''}2^{n^\alpha}(n^\alpha + 2)$. Hence, for every $c' < c$, we have $\varepsilon(n) < 2^{-c'n^\delta}$ and $t(n) > 2^{c'n^\delta}$ for large enough n. □

Combining Proposition 4.6 with Theorem 4.4 yields the following result.

Corollary 4.9. *Let $s \geq 0$, z, $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$, $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be as defined in Theorem 3.8, then the oracle-aided function ensemble $\mathcal{ADW}_z(\mathcal{H}, \mathcal{G}, \mathsf{GGM}_m^G)$ is a $(m(n), q(n), t_G(n) - p(n) \cdot m(n) \cdot q(n), 2m(n) \cdot q(n) \cdot \varepsilon_G(n) + 1/q(n)^{s+1})$-PRG-to-PRF reduction, where $p \in$ poly is determined by the evaluation and sampling time of $\mathcal{H}$, $\mathcal{G}$ and G, and $q(n) \leq 2^{m(n)}/(1+\zeta)$ where ζ is from Theorem 3.8.*

4.1.1 Settling the Parameters

In the construction given in Corollary 4.8, assuming $2^{n^{1/2}} \leq q < 2^n$, we need to set k to be $\Theta(n)$. Higher independence means that we need to use a longer key and the evaluation time is larger. More accurately, the evaluation time of our construction is lower bounded by the evaluation time of the $\Theta(n)$-wise independent hash function (that may or may not be larger than $\Theta(\log(q) \cdot e_G)$). Moreover, the key length must be $\Theta(n^2)$, but this can be circumvented by extending a short key to a long one using GGM. This costs additional $\Theta(n \cdot e_G)$ time. As we have stated in Corollary 4.8, our construction gives a $(O(n^\alpha), 2^{n^\alpha}, 2^{c'n^\delta}, 2^{c'n^\delta})$-PRG-to-PRF reduction for every $0 < c' < c$, and works for any $0 < \alpha < \delta$, while the construction of JPT only works for $\delta/2 \leq \alpha < \delta$.

In the construction given in Corollary 4.9, assuming $2^{n^{1/2}} \leq q < 2^n$, we can get better results (that also improve upon Jain et al. [20]). Assume for simplicity of the exposition that $q = 2^{n^{1/2}}$. As opposed to the specification of parameters in Theorem 3.8, we can use long random tables of size $\Theta(\sqrt{q})$ (i.e., $\delta = 1/2$ in Theorem 3.8), $s = n/\log q$ and get that setting $k = \Theta(n^{1/2})$ is enough to get error $O(1/2^n)$. The point is that the creation of the (long) tables that we need in Theorem 3.8 (which so far were part of the key) can be done by applying GGM on a short input key. This only increases the evaluation time by an additional $\Theta(\log(q) \cdot e_G)$ term. In total, the evaluation time of our construction is $\Theta(\log(q) \cdot e_G)$.

5 From Non-Adaptive to Adaptive PRF

In this section we show how to apply the Cuckoo Hashing based constructions of Section 3 in order to come up with a construction of an adaptive PRF from non-adaptive one in a security preserving manner. As in Section 4, let $\mathcal{P}(\mathcal{U},\mathcal{V}) = \{\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n)\}_{n\in\mathbb{N}}$ be the ensemble of function families $\{P_{u,v}\colon \mathcal{D}_n \mapsto \mathcal{R}_n\}_{(u,v)\in\mathcal{U}_n\times\mathcal{V}_n}$, where $\mathcal{U}_n$ and $\mathcal{V}_n$ are some non-empty sets as described in Section 3. We begin by showing that $\mathcal{P}(\mathcal{U},\mathcal{V}^{\mathcal{F}})$ is computationally indistinguishable from $\mathcal{P}(\mathcal{U},\mathcal{V}^{\Pi})$, where $\mathcal{V}^{\mathcal{F}}$ denotes that $\mathcal{V}$ is implemented using *non-adaptive* pseudorandom functions and $\mathcal{V}^{\Pi}$ denotes that $\mathcal{V}$ is implemented using truly random functions (see also Footnote 14). In the sequel, we assume that $\mathcal{V}$ is implemented using 2 calls to the underlying functions (as this is the case in the actual constructions we work with).

For ease of notation, in the following we assume $\ell(n) = n$ (i.e., $\mathcal{F}$ is length preserving).

Lemma 5.1. *Let $\mathcal{P}(\mathcal{U},\mathcal{V}) = \{\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n)\}_{n\in\mathbb{N}}$ be the ensemble of function families $\{P_{u,v}\colon \mathcal{D}_n \mapsto \mathcal{R}_n\}_{(u,v)\in\mathcal{U}_n\times\mathcal{V}_n}$, that is implementation of $\mathcal{PP}$ or $\mathcal{ADW}$, where $\mathcal{U}_n$ and $\mathcal{V}_n$ are arbitrary sets, $\Pi = \{\Pi_n\colon \mathcal{D}_n \mapsto \mathcal{R}_n\}_{n\in\mathbb{N}}$, where Π_n is the set of all functions from $\mathcal{D}_n$ to $\mathcal{R}_n$, and $\mathcal{F} = \{\mathcal{F}_n\colon \mathcal{D}_n \mapsto \mathcal{R}_n\}$ be an efficient function family. Let $\zeta > 0$ that depends on the instantiation of $\mathcal{P}$.[17] Then for every $q = q(n)$-query, oracle-aided, adaptive distinguisher D of running time $t = t(n)$, there exists a $(1+\zeta)q$-query, non-adaptive, oracle-aided distinguisher $\widehat{\mathsf{D}}$ of running time $p(n)t(n)$, with*

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

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

Proof. The proof follows along similar lines to the proof of [6, Lemma 1]. Let $\widetilde{\mathsf{D}}$ be the following two-oracle distinguisher:

Algorithm 5.2 ($\widetilde{\mathsf{D}}$)**.**

Input: 1^n.

Oracles: *Functions ϕ_1 and ϕ_2 from $\mathcal{D}_n$ from $\mathcal{R}_n$.*

1. *Compute $\phi_1(x)$ and $\phi_2(x)$ for every $x\in[(1+\zeta)q(n)]_{\mathcal{D}_n}$.*
2. *Set f according to $\mathcal{P} = (\mathcal{U}_n,\mathcal{V}_n^{\phi_1,\phi_2})$.*
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 $\widetilde{\mathsf{D}}$ makes $(1+\zeta)q(n)$ *non-adaptive* queries to ϕ_1 and ϕ_2, and it can be implemented to run in time $t(n)e_{\mathcal{P}}(n) + 2(1+\zeta)q(n) + e_q(n) \leq r(n)t(n)$ (where $e_{\mathcal{P}}$ and e_q are the sampling and evaluation time of $\mathcal{P}$ and q respectively) for large enough $r\in$ poly. Observe that both possible implementations of $\mathcal{P}$, i.e., $\mathcal{PP}$ or $\mathcal{ADW}$, only query ϕ_1 and ϕ_2 on the first $(1+\zeta)q(n)$ elements of their domain. Hence, in case ϕ_1 and ϕ_2 are uniformly drawn from $\mathcal{F}_n$, then the emulation of D

[17] In case we use $\mathcal{PP}$ function family (Definition 3.3) then $\zeta = 3$, and in case we use $\mathcal{ADW}$ function family (Definition 3.7) then ζ is of Theorem 3.8.

done in $\widehat{\mathsf{D}}^{\phi_1,\phi_2}$ is identical to a random execution of D^f with $f \leftarrow \mathcal{P}(\mathcal{U}_n, \mathcal{V}_n^{\mathcal{F}_n})$. Similarly, in case ϕ_1 and ϕ_2 are uniformly drawn from Π_n, then the emulation is identical to a random execution of D^f with $f \leftarrow \mathcal{P}(\mathcal{U}_n, \mathcal{V}_n^{\Pi_n})$. Thus,

$$\begin{aligned}&\left|\Pr_{(f_1,f_2)\leftarrow\mathcal{F}_n\times\mathcal{F}_n}[\widetilde{\mathsf{D}}^{f_1,f_2}(1^n)=1]-\Pr_{(\pi_1,\pi_2)\leftarrow\Pi_n\times\Pi_n}[\widetilde{\mathsf{D}}^{\pi_1,\pi_2}(1^n)=1]\right| \\ &=\left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\mathcal{F}_n})}[\mathsf{D}^f(1^n)=1]-\Pr_{f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\Pi_n})}[\mathsf{D}^f(1^n)=1]\right|.\end{aligned} \tag{7}$$

Hence, Lemma 2.8 yields that there exists a $(1+\zeta)q(n)$-query, non-adaptive, single-oracle distinguisher $\widehat{\mathsf{D}}$ of running time $r(n)t(n)+2q(n)e_{\mathcal{F}}(n) \leq p(n)t(n)$, where $p \in \text{poly}$, such that

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

□

Let $\mathcal{D}_n = \mathcal{R}_n = \{0,1\}^n$, $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be a non-adaptive $((1+\zeta)q(n), p(n)t(n), \varepsilon(n))$-PRF and let D be an oracle-aided algorithm of running time $t(n)$ making at most $q(n)$ queries. Lemma 5.1 yields that for large enough n

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

Assuming the conditions of Lemma 3.2 with respect to $\mathcal{P}(\mathcal{U},\mathcal{V})$ hold, we get that for some ε'

$$\left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\Pi_n})}[\mathsf{D}^f(1^n)=1]-\Pr_{\pi\leftarrow\Pi_n}[\mathsf{D}^{\pi}(1^n)=1]\right| \leq \varepsilon'(n) \tag{8}$$

for every $n \in \mathbb{N}$. Hence, by the triangle inequality

$$\left|\Pr_{f\leftarrow\mathcal{P}(\mathcal{U}_n,\mathcal{V}_n^{\mathcal{F}_n})}[\mathsf{D}^f(1^n)=1]-\Pr_{\pi\leftarrow\Pi_n}[\mathsf{D}^{\pi}(1^n)=1]\right| \leq 2\varepsilon(n)+\varepsilon'(n).$$

Plugging in Lemma 3.5 we immediately get the following theorem.

Theorem 5.3. *Let q be a polynomial-time computable integer function, 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 $(c\log(q(n)))$-wise independent function family ensembles, where $c>0$ is universal, and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be a non-adaptive $(4q(n), p(n)t(n), \varepsilon(n))$-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 $(q(n), t(n), 2\varepsilon(n)+1/q(n))$-PRF.*

Proof. Recall that in this case $\varepsilon'(n) = q(n)/2^{\Omega(c\log(q(n)))}$. Setting c such that $q(n)/2^{\Omega(c\log(q(n)))} = 1/q(n)$ complete the proof. □

Theorem 5.3 yields the following simpler corollary.

Corollary 5.4 (Restatement of Theorem 1.4)**.** *Let q, $\mathcal{H}$, $\mathcal{G}$ and p be as in Theorem 5.3. Assuming $\mathcal{F}$ is a non-adaptive $(p(n)t(n))$-PRF, then $\mathcal{PP}(\mathcal{H},\mathcal{G},\mathcal{F})$ is an adaptive $(t(n)/4)$-PRF.*

Plugging in Lemma 3.9 we immediately get the following (similarly to Theorem 5.3):

Theorem 5.5. *Let q be a polynomial-time computable integer function, let $s \geq 0$, z, $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto [(1+\zeta)q(n)]\}$, $\mathcal{G} = \{\mathcal{G}_n \colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be as defined in Theorem 3.8, and let $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ be a non-adaptive $((1+\zeta)q(n), p(n)t(n), \varepsilon(n))$-PRF, where $p \in$ poly is determined by the evaluation time of $q, \mathcal{H}, \mathcal{G}$ and $\mathcal{F}$. Then $\mathcal{ADW}_z(\mathcal{H}, \mathcal{G}, \mathcal{F})$ is an adaptive $(q(n), t(n), 2\varepsilon(n) + 1/q(n)^{s+1})$-PRF.*

Notice that in Theorem 5.5 we used the setting of parameters of Theorem 3.8. In particular, since the matrices that are sampled in the definition of $\mathcal{ADW}_z$ are of constant size, we can embed them into the key of the PRF.

6 Further Research

The focus of this paper was on PRFs. In Sections 4 and 5 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 goind 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. One related paper is that of Hoang et al. [19], that gives a method to convert a PRF into a PRP with beyond-birthday security. Another related work is of Håstad [17] 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 [13] 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 [28] 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 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 [6] in a non security preserving manner.

Recently Pătraşcu and Thorup [42] 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.

7 Acknowledgments

We thank Eylon Yogev and the anonymous referees for their helpful comments. The third author would like to thank his 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. 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.

[5] 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.

[6] I. Berman and I. Haitner. From non-adaptive to adaptive pseudorandom functions. In *Theory of Cryptography, 9th Theory of Cryptography Conference, TCC 2012*, pages 357–368, 2012.

[7] 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.

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

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

[10] N. Chandran and S. Garg. Hardness preserving constructions of pseudorandom functions, revisited. *IACR Cryptology ePrint Archive*, 2012:616, 2012.

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

[12] 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.

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

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

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

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

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

[18] 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.

[19] 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.

[20] 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.

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

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

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

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

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

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

[27] 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.

[28] 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.

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

[30] 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.

[31] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of psuedo-random functions. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 170–181, 1995.

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

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

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

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

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

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

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

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

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

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

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

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

[44] 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 Adaptive Security of Functions Family – A General Framework

We provide a general framework for proving the *adaptive* security of functions families, given that the analysis of the security of the *non-adaptive* case behaves in a certain way: we need to be able to separate the 'key' into two parts, drawn from $\mathcal{U}$ and $\mathcal{V}$ respectively. Then the anlysis has to guarantee that provided the *non-adaptive* distinguisher did not stumble upon a BAD set in $\mathcal{U}$, the randomness of $\mathcal{V}$ should make his view completely random. If the analysis is of this nature, then we claim that the resulting construction is secure against adaptive attacks as well.

Lemma A.1 (Restating Lemma 3.2)**.** *Let $\mathcal{F} = \mathcal{F}(\mathcal{U}, \mathcal{V})$ be a function family of the form $\{f_{u,v} \colon \mathcal{D} \mapsto \mathcal{R}\}_{(u,v)\in\mathcal{U}\times\mathcal{V}}$, where $\mathcal{U}$ and $\mathcal{V}$ are arbitrary non-empty sets, let $t \in \mathbb{N}$ and let $\mathrm{BAD} \subseteq \mathcal{D}^{\leq t} \times \mathcal{U}$ be a left-monotone set. Assume that for every $\overline{q} \in \mathcal{D}^{\leq t}$ it holds that*

1. $\left(f(\overline{q}_1), \ldots, f(\overline{q}_{|\overline{q}|})\right)_{f\leftarrow\{f_{u,v}\colon v\in\mathcal{V}\}}$ *is uniform over $\mathcal{R}^{|\overline{q}|}$ for every $u \in \mathcal{U}$ such that $(\overline{q}, u) \notin \mathrm{BAD}$, and*
2. $\Pr_{u\leftarrow\mathcal{U}}[(\overline{q}, u) \in \mathrm{BAD}] \leq \varepsilon$,

then for any t-query adaptive *algorithm* D*, it holds that*

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

where Π is 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 define an Algorithm A.2 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 $\overline{v}_{1,\ldots,0} = \lambda$, where λ is the empty vector. Consider the following random process:

Algorithm A.2.

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 $(\lambda, u) \in \mathrm{BAD}$, set $v \leftarrow \mathcal{V}$.*

4. *Emulate* D *again, while answering the* i^{th} *query* q'_i *according to the following procedure:*
   (a) *If* $(\overline{q}'_{1,\ldots,i} = (q'_1, \ldots, q'_i), u) \notin \mathrm{BAD}$, *answer with* $a'_i = a_i$ *(the same* a_i *from Step 1).*
   (b) *Otherwise* $((\overline{q}'_{1,\ldots,i}, u) \in \mathrm{BAD})$*:*
      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* $a'_i = f_{u,v}(q'_i)$.
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.2. 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.3 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.2 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

$$\Pr[\overline{A} \neq \overline{A}'] \leq \Pr[(\overline{Q}, U) \in \mathrm{BAD}] \tag{9}$$

In addition, since U is chosen *after* $\overline{Q}$, the second condition of Lemma A.1 yields that

$$\Pr[(\overline{Q}, U) \in \mathrm{BAD}] \leq \varepsilon \tag{10}$$

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.3. $\overline{A}'$ *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.2 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., $k+1$] queires 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.2 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.4. *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.4. We prove by reverse induction on $i = |\overline{w}|$. For the base case $i = t$, we note that (by definition) Algorithm A.2 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 $(\overline{q}^{+}_{\overline{w}}, u) \in \text{BAD}$, Algorithm A.2 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 $(\overline{q}^{+}_{\overline{w}}, u) \notin \text{BAD}$.

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

$$\frac{|\mathcal{S}_{\overline{w}'}|}{|\mathcal{V}|} = \Pr_{v \leftarrow \mathcal{V}}[v \in \mathcal{S}_{\overline{w}'}] = \Pr_{v \leftarrow \mathcal{V}}[f_{u,v}(\overline{q}_{\overline{w}'}) = \overline{w}'] = \frac{1}{|\mathcal{R}|^{|\overline{w}'|}}, \tag{11}$$

for every $\overline{w}' \in W$, where the third equality of Equation (11) holds by the first property of $\mathcal{F}(\mathcal{U}, \mathcal{V})$ (as stated in Lemma A.1). □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de